INSTITUTO POLITÉCNICO DE BEJA

Escola Superior de Tecnologia e Gestão

Mestrado em Engenharia de Segurança Informática

Plataforma Parametrizável para Análise Forense de Dispositivos Móveis

Análise Forense para SO Android

Francisco Nicolau Gomes Chainho

2014

INSTITUTO POLITÉCNICO DE BEJA

Escola Superior de Tecnologia e Gestão

Mestrado em Engenharia de Segurança Informática

Plataforma Parametrizável para Análise Forense de Dispositivos Móveis

Análise Forense para SO Android

Elaborado por:

Francisco Nicolau Gomes Chainho

Orientado por:

Professor Doutor Rui Miguel Soares Silva

2014

Resumo

Plataforma Parametrizável para Análise Forense de Dispositivos Móveis

Análise Forense para SO Android

A Computação Forense tem procurado dar reposta as questões dos investigadores em diversos sistemas informaticos, sempre com o objetivo de procurar evidências para recriar a verdade de um evento. Esta dissertação centra-se na Computação Forense em SO *Android*. Procurando em primeiro lugar aplicar uma metodologia a investigação dos SO *Android* e em segundo lugar implementar um *software* que consiga extrair dados de um dispositivos móvel. Sempre com o objetivo de que esse *software* funcione em qualquer versão o SO *Android* e consiga gerar relatórios de forma automatizada. Para testar o sistema implementado são feitos testes de extração em dois dispositivos móveis, com diferentes versões do SO *Android* para comprovar a eficácia do sistema e gerar um relatório com os dados extraídos.

Palavras-chave: *Android, Forensics, Agente Forense, DroidExport, DroidImport, Metodologias de Investigação, Extração de Dados, Dispositivos Móveis.*

Abstract

Parameterizable Forensic Analysis Platform for Mobile Devices

Forensic Analysis for Android OS

The Computer Forensics seeks to answers the questions of researchers, in computer systems always with the aim of looking for evidence to recreate the truth of an event. This dissertation focuses on Computer Forensics in Android OS. Looking firstly apply a research methodology to Android OS and secondly implement a software that can extract data from a mobile device. Always with the aim that this software works on any version of the Android OS, and achieving to generate automated reports. To test the implemented system these are made extraction on two mobile devices, with different versions of Android OS to prove the efficacy of the system and generate a report with the extracted data.

Keywords: *Android, Forensics, Forensic Agent, DroidExport, DroidImport, Research Methodologies, Data Extraction, Mobile Devices.*

Agradecimentos

A todos os que me ajudaram ao logo deste percurso que contribuiu para que conseguisse concluir esta da fase da minha vida académica.

Ao Professor Dr. Rui Miguel Silva pela sua orientação e visão critica que me concedeu durante a elaboração da Dissertação de Mestrado.

Ao Prof. Manuel David Masseno pelo seu esclarecimento a nível legislativo.

Índice Geral

Índice de Figuras

Índice de Tabelas

Índice de Listagens

Abreviaturas e Siglas

ACOP - Association of Chief Police Officers Open Handset Alliance
ADB - Android Debug Bridge
ADT - Android Development Tools
API - Application Programming Interface
ARP - Address Resolution Protocol
ADEL - Android Data Extractor Lite
BBM - Balckberrrymessenger
CDMA - Code Division Multiple Access
CRMI - Custom Recovery Mode Image
CVS - Concurrent Version System
DDS - Deployable Device Seizure
DFRWS - Digital Forensics Research Workshop
DVM - Dalvik Virtual Machine
EDGE - Enhanced Data rates for GSM Evolution
ESN - Electronic Signage Network
eMMC - Embedded MultiMediaCard
EXT4 - Fourth Extend file System
GB - Gigabyte
GPS - Global Positioning System
GSM - Global System for Mobile
HTTP - Hypertext Transfer Protocol
IDE - Integrated Development Environment
IP - Internet Protocol
IDEN - Integrated Digital Enhanced Network
IMSI - International mobile subscriber identity
IMEI - International Mobile Equipment Identity
JAVA - Just Another Virtual Architecture
JRE - Java Runtime Environment
LSASS - Security Authority Subsystem Service Local

MD5 - Message-Digest algorithm 5
MEID - Mobile Equipment Identifier
MMS - Multimedia Messaging Service
MRU - Most Recently Used
NFC - Near Field Communication
NIST - National Institute of Standards and Technology
PDA - Personal digital assistant
PDF - Portable document format
PIN - Password Identification Number
SD - Storage Device
SDK - Software Developmemt Kit
SHA-1 - Secure Hashing Algorithm 1
SIM - Subscriber Identity Module
SMS - Short Message Service
SPN - Service Provider Name
SQL - Structured Query Language
TDMA - Time Division Multiple Access
UE - União Europeia
USB - Universal Serial Bus
URL - Uniform Resource Locator
WiFI - Wireless Fidelity
XML - Extensible Markup Language
YAFFS2 - Yet Another Flash File System 2

Capítulo 1

Introdução

Com a evolução dos Sistema Comunicação e dos Sistemas Informáticos, que cada vez mais fazem parte integrante da nossa vida, facilitando as nossas ações em diversos aspetos do nosso dia-a-dia, verificamos que são poucas as situações em que não temos sistemas informáticos para nos facilitar as tarefas. Mas toda esta evolução traz um lado menos positivo, surgindo novas formas de cometer crimes, porque a evolução do conhecimento também se verifica do lado de quem pretende utilizar os sistemas informáticos para cometer crimes informáticos. A crescente ocorrência dos mesmos deu origem à necessidade de criar uma forma legal de punir estes atos. Dando origem à investigação forense aplicada aos sistemas informáticos com o objetivo de combater os crimes informáticos, atuando também como prevenção criminal.

Esses avanços tecnológicos tiveram forte implicação no quotidiano de todos, nomeadamente no telefone móveis, tanto a nível de hardware, *software* e de sistema de comunicação, o que originou um novo conceito de dispositivo móvel, o *smartphone*. Telefones móveis com grandes capacidade de processamento, o que aumentou exponencialmente os recursos aos utilizadores, sendo possivel num dispositivo ter como o acesso a *Internet*, tirar fotografias, fazer vídeo, ter acesso a um conjunto de aplicações desenvolvidas para dispositivos móveis, entre outras. Com essa evolução os utilizadores passaram a dar outra utilização aos dispositivos móveis, o que implica que vão conter outro tipo de informações para além das que existe num telefone móvel, desde fotografias, acessos *web*, *email*, dados das aplicações, entre outros. Os dispositivos móveis tornaram-se pequenos computadores que acedem e permitem produzir nos diversos formatos de ficheiros uma grande quantidade de informação.

Os dispositivos móveis armazenam informação do utilizador, no caso de ser necessário fazer uma investigação forense ao dispositivo, essa informação torna-seimportante. Sendo necessário extrair todos os dados do dispositivos e aqui começam

as dificuldades. A realização de uma analise forense pode dar muitas informações sobre o utilizador, sendo uma ferramenta de comunicação tem informação armazenada sobre os registo de chamadas, de mensagens de texto e imagem, tráfego de *internet*, agenda, contactos, acesso a dados na *cloud*, *email*, entre outros.

Estando o investigador por vezes sujeito a algumas restrições técnicas quando tem de realizar uma investigação, nem sempre é possível extrair essa informação, um exemplo, quando o dispositivo não permite o acesso via USB (*Universal Serial Bus*), o investigador tem para isso que utilizar uma metodologia de investigação, que permita obter o maior numero de dados com base no estado do dispositivos e nas restrições que encontrar ao longo do processo.

O sistema operativo que lidera nos dispositivos móveis é o *Android*, um sistema de *Open Source* que consegue se adaptar a vários tipos de dispositivos e que tem evoluído no sentido de fornecer cada vez mais funcionalidade aos utilizadores. Sendo o SO (Sistema Operativo) mais popular, ele é o foco do trabalho desenvolvido ao longo desta dissertação. Os dispositivos móveis têm um grande dinamismo a que se junta uma enorme diversidade de modelos. Equipados com o SO *Android* que têm várias versões, adaptadas para a cada dispositivo pelas empresas que os fabricam, principalmente nos equipamentos de baixo custo, devido as alterações que são introduzidas no SO para se adaptarem ao dispositivo, o que dificulta a definição de padrões de análise forense para os SO resultando num problema para os investigadores forenses, tornando-se assim difícil desenvolver um único procedimento ou ferramentas que sejam capazes de realizar uma análise a todos pontos importantes do dispositivo.

Os objetivos principais do trabalho consistem em implementar um agente forense que permita extrair informação do dispositivo independente da versão do SO, dando também importancia a metodologia que se utiliza para para realizar investigação, bem como o estudo do estado da arte.

Neste primeiro capitulo foi apresentada uma breve introdução ao tema, onde se estabeleceu uma ligação evolutiva entre sistema informativos, investigação forense, dispositivos móveis e o SO *Android*. Descrevendo os objetivos que se pretendem alcançar na elaboração desta dissertação.

No Capítulo 2, pretende-se dar uma visão da Computação Forense aplicada a computadores e informação volátil, as varias definições que se podem encontrar e os principais sistemas onde se aplica a investigação forense. São referidos dois dos desafios que se apresentam aos investigadores. É também apresentado um modelo de computação forense aplicável pelos investigadores aos sistemas informativos. Este capitulo serve como uma introdução geral ao tema da Computação Forense antes

de se focar no caso especifico dos dispositivos móveis.

No Capítulo 3, descreve-se a plataforma *Android*, as suas principais características, versões e funcionalidades, pretende-se dar a conhecer os pontos principais do SO a ter em conta ao desenvolver a investigação. Ao nível da computação forense para dispositivos móveis são apresentadas características da investigação forense e uma metodologia de investigação que se aplica ao SO *Android*.

No Capítulo 4, elabora-se uma analise ao Estado da Arte. Focando quantro pontos principais, Independência do tipo de plataforma *Android*, Metodologia de analise, *Software* de Analise Forense e Análise dos dados extraídos. Descreve-se também a formulação da hipótese de investigação com base na aplicação da metodologia de investigação do Capítulo 3 ao caso em estudo.

No Capítulo 5, descreve-se a implementação do Agente Forense. Sendo analisada a Arquitetura do Sistema, as Classes e Métodos e Modelo de Comunicação.

No Capítulo 6, realizam-se testes utilizando o Agente Forense em vários cenários. São também apresentados alguns *software* de extração de dados que podem concorrer com o Agente Forense desenvolvido.

No Capítulo 7, conclui-se o trabalho, sendo apresentado uma síntese dos resultados obtidos atraves da aplicação da metodologia aos cenários em análise. São propostos os trabalhos futuros ao nível de novas funcionalidades a incluir no agente forense, de forma a dar continuidade ao trabalho desenvolvido.

Desta forma, e como se pode verificar a dissertação foi estruturada em sete capítulo, através do quais se descreve como se antingiu o objetivo proposto, ao implementar um agente forense que permite extrair informação do dispositivo independente da versão do SO *Android*.

Capítulo 2

Computação Forense

2.1 Introdução Genérica

Computação Forense, dividindo o termo obtemos Computação que definimos como um conjunto de conhecimentos e técnicas referentes à utilização do computador ou processamento de dados e Forense que definimos como o processo de utilizar o conhecimento científico para a recolha e análise de provas com a finalidade de serem apresentadas em tribunal, dado que forense significa "referente ao foro judicial"[1].

A computação forense surge da necessidade de padronizar a forma de apresentação de provas digitais em tribunal, avançando assim, para a era das novas tecnologias. Para um termo mais correcto, definimos a Computação Forense como a disciplina que combina elementos de direito e ciências da computação para a recolha e análise de dados de sistemas de computadores, redes, comunicações sem fio e dispositivos de armazenamento de um modo admissível, como prova num tribunal.[2]

Uma das definições que se encontra para computação forense, foi criada na DFRWS (*DigitalForensics Research Workshop*) em 2001, onde consta que: A utilização de métodos comprovados cientificamente vocacionados para a preservação, recolha, validação, identificação, análise, interpretação, documentação e apresentação das evidências digitais com o propósito de facilitar e promover a reconstrução de eventos que tenham tido fins criminosos, ou ajudar a antecipar ações não autorizadas que possam ser prejudiciais para operações planeadas.[3]

O contexto de utilização de técnicas de computação forense é varidado, não sendo apenas aplicados em processos de investigação criminal, embora os princípios e os

procedimentos sejam praticamente os mesmos, tendo como base o conceito de investigação de "algo". O tipo de investigação pode ser variado, mas tem uma base comum, informação digital, mesmo quando retirada de equipamentos e fontes diferentes, nomeadamente computadores, *smartphone*, *tablet* ou redes de computadores.

Uma das características do termo computação forense é ser descrito como sendo uma arte e uma ciência. Na visão de *Wietse Venema e Dan Farmer*, defendem que por vezes o examinador age como um "arqueólogo" digital onde procura a actividade do utilizador, o conteúdo dos ficheiros, o tempo de acesso, a informação sobre ficheiros apagados e os fluxos de rede, outras vezes o examinador age como um "geólogo" digital onde se procura perceber os processos automáticos que o utilizador não tem controlo direto, como os numero de processos, os dados em memória, a localização dos dados em disco.[4]

Uma das sugestões que existe para a definição, é de que a dicotomia entre a arte e ciência da análise forense não é um paradoxo, mas simplesmente uma aparente inconsistência decorrente da fusão de dois aspetos práticos: a ciência forense combinada com a arte da investigação. Aplicando o método científico e o raciocínio dedutivo aos dados é a ciência, a interpretação desses dados para reconstruir um evento, é a arte.[4]

Uma investigação forense digital é um caso especial de uma investigação forense, onde os procedimentos e técnicas que são utilizadas permitem que os resultados obtidos sejam credíveis de serem utilizados num processo judicial. É necessário que tenham sido tomadas medidas para preservar o estado do computador através de ferramentas apropriadas.

Onde se aplica

O objetivo da realização de um exame forense é procurar evidências para recriar a verdade de um evento. Cada exame começa com a formulação de uma hipótese, por exemplo "computador utilizado para fazer acessos ilegais a servidores de Entidades Bancárias", "foram eliminados ficheiros do disco", há que procurar evidências que comprovem ou não essa hipótese. Evidências digitais são os dados que suportam ou refutam a hipótese que foi formulada durante a investigação. Esta é uma noção geral de provas e podem incluir dados que sejam admissíveis num tribunal, devido à forma como são obtidos, de forma legal ou não [5]. Todas as ações que se efetuam num sistema informático deixam evidências no sistema, independentemente do tipo de ações que são efetuadas todas elas deixam registos que com maior ou menor

dificuldade podem ser encontradas pelo examinador.

2.2 Análise Forense de Sistemas Informáticos

Os procedimentos básicos de cada investigação visam atuar em quatro áreas, são a aquisição, as evidências, a análise das evidências e o relatório. A figura 2.1 demonstra de forma simples a interligação das varias áreas.

Figura 2.1: Áreas de Análise Forense

- Aquisição: O processo de aquisição de provas digitais tem de ser feito de modo a preservar os dados em que se mantenha a cadeia de custódia (A cadeia de custódia contribui para a validação da prova pericial e do relatório produzido). Os dados recolhidos contêm informações necessárias para se perceber como os eventos ocorreram e como os recursos ou dados possam ter sido afetados.
- Evidências: Toda a recolha deve ser feita de forma a manter a integridade das provas. Efetuar-se uma cópia de todos os dados e será sobre essa cópia que é feita a análise forense desta forma garante-se que a prova original não é alterada.
- Análise: É a identificação de ações não autorizadas ou anómalas que existam, como foram implantadas e os acesso que poderiam ter tido ao sistema, entre outros registos. Depois de identificar algo que posso confirmar as suspeitas, o próximo passo é determinar as consequências dos eventos e avalia-los em conformidade com a legislação, bem como o impacto que tiveram.
- Relatórios: Todos os detalhes e processos utilizados são documentados pelos investigadores assim que se começa a delinear a investigação, será um processo

contínuo até a obtenção de uma conclusão. Cada procedimento da metodologia definida é registado para facilitar a validação da investigação. Todo o conteúdo terá de respeitar os dados relativos à investigação nomeadamente, informações confidências, pessoais, dados sobre vulnerabilidades ou informações que possam ser necessárias para a aplicação da lei em vigor [6] [7].

É importante que todos os processos utilizados para obter e analisar dados, realizados durante a investigação não os alterem, a fim de poderem ser utilizados como prova admissível em tribunal.

Para garantir a autenticidade das evidências, os investigadores utilizam uma técnica para certificar o conteúdo original do dispositivo e a cópia que se fez para análise. Esta certificação funciona com uma impressão digital eletrónica, para isso gerar-se uma tabela *hash*, que analisa todos os *bit* que existem nos dados e cria o *hash* correspondente, qualquer alteração a um desses *bit* vai dar origem a um *hash* diferente. Assim os *hash* dos dados originais e da cópia para serem validados terão de ter valores *hash* idênticos. O padrão para a criação de *hash* aconselha que se utilize o algoritmo MD5 (*Message-Digest algorithm 5*) criado por *Ronal L. Rivest*, que o descreve como um algoritmo utilizado para verificar a integridade dos dados por meio da criação de mensagem de 128 *bits* a partir dos dados de entrada, que têm por base a geração de um valor único para esses dados, criando uma assinatura digital. O algoritmo é destinado a aplicações de assinaturas digitais onde os ficheiros de grandes dimensões podem ser crifrados com uma chave privada sob um sistema de criptografia de chave pública RSA [8].

2.2.1 Sistemas *Linux*

O analista deve compreender e conhecer bem o funcionamento do SO *Linux* para perceber onde tem de procurar a informação, como os ficheiros estão organizados e como são geridos. A estrutura do disco possui pelo menos três tipo de partições, *home*, *swap*, /, para perceber como os dados são guardados pelo sistema é necessário entender como e quais são as partições utilizadas. É importante também conhecer como funciona a memória física e a *swap* para perceber como capturar e analisar os dados.

O *Linux* visualiza todos os sistemas de ficheiros na perspetiva de um conjunto de objetos comum. Esses objetos são *superblock*, *inode*, *dentry* e *file*. Na raiz de cada sistema de ficheiros está *superblock*, que descreve e mantém o estado do sistema de ficheiros. Cada objeto que é gerido dentro de um sistema de ficheiros (ficheiro ou

diretoria) é representado no *Linux* como um *inode*. O *inode* contém todos os metadados para gerir os objetos do sistema de ficheiros que incluem o ID do proprietário, permissões (ler, escrever, executar), número de ligações, MAC *time* da última modificação, acesso e alteração de estatuto (alteração de proprietário, permissão ou o número de links) e dimensão do ficheiro. Outro conjunto de estruturas, definidas por dentries, é utilizado para conversão entre nomes e *inode*s, neste conjunto a diretoria mantém em *cache* o ultimo objeto [9]. O sistema de diretorias e ficheiros do *Linux* é dividido entre as pastas: *bin,boot, dev, etc, home, lib, mnt, media, opt, proc, root, sbin, tmp, usr e var*. [10].

Tabela 2.1: Diretorias do SO *Linux*

Diretoria	**Descrição**	**Diretoria**	**Descrição**
/	Raiz do sistema de ficheiro	/*bin*	Ficheiros binários do sistema
/*boot*	Ficheiro de inicialização do sistema	/*dev*	Repositório do sistema
/*var*	Ficheiros e sistema e logs do sistema	/*home*	Diretoria de ficheiros o utilizador do sistema
/*lib*	Biblioteca de executaveis	/*mnt*	Diretoria de montagem de Discos
/*media*	Diretoria de montagem de discos	/*opt*	Diretoria de instalação de alguns programas
/*proc*	Diretoria de processamento	/*root*	Diretoria dos ficheiros do administrador do sistema
/*sbin*	Mantém a máquina de utilizador *root*	/*tmp*	Diretoria de ficheiros temporarios
/*usr*	Ficheiros partilhados do sistema		

O SO *Linux* funciona na lógica de tudo ser um ficheiro, pasta, documento, diretoria. Tudo é considerado um ficheiro, tendo um sistema de proteção que funciona pela atribuição de permissões de acesso. Divididos por permissões para utilizadores e permissões para ficheiros e diretorias [10].

O sistema de ficheiros do *Linux* gere os dados em blocos. Os ficheiros que são excluídos são marcados na diretoria de entrada, assim o nome do ficheiro é referenciando como não estando a ser utilizado. O *inode* dos ficheiros é marcado como não utilizado mantendo só alguns dos seus atributos. Os blocos de dados são marcados como não utilizados. Por vezes os ficheiros apagados são mantidos pelo sistema durante muito tempo devido a boa gestão do SO Que agrupa os ficheiros em

Tabela 2.2: Permissões *Linux*

Permissões	**Descrição**
UID	Identificação do utilizador ou dono do ficheiro
GID	Identificação do grupo
OUTROS	Categoria dos utilizadores que não são donos dos ficheiros ou diretoria

Tabela 2.3: Permissões Utilizadores *Linux*

Permissões	**Descrição**	**Permissões**	**Descrição**
r	Pode ler o ficheiro	w	Permissão de gravação para ficheiros ou diretoria
x	Permissão para executar ficheiros	d	O ficheiro é uma pasta
1	O ficheiro é um *link*	“ “	É um ficheiro comum
-rwx— —	Permissões de acesso do dono do ficheiro	- —rwx—	Refere-se às permissões de acesso do grupo do ficheiro
- — —rwx		0	
1	acesso de outros utilizadores ao ficheiro	2	Permissão de gravação(igual a +w)
3	Permissão do tipo 1 refere-se a execução do ficheiro (igual a +x)	4	Permissão de leitura (igual a +r)
5	Permissão de leitura e execução (igual a +rx)	6	Permissão de leitura e gravação (igual a +rw)
7	Todas as permissões (igual a +rwx)	u	Utilizador
g	Grupo	o	Outros
a	Todos		

vez de os colocar aleatoriamente, evitando assim a sua fragmentação e facilitando a sua recuperação.

Um dos atributos importantes para investigadores são os MAC *time* que registam o último instante em que ocorreram certos eventos pertinentes a um dado ficheiro. Todas as alterações que se façam nos ficheiros deixam um registo permitindo aos analistas perceber quais foram as alterações efetuadas.

No *Linux* é possível repetir os comandos utilizados em sessões anteriores, para isso os comandos são guardados num ficheiro de histórico de *shell*. Assim é possível investigar os comandos que foram executados por um utilizador em sessões anteriores de forma a perceber quais foram as ações realizadas. Uma das principais fontes de

informação são os ficheiros de *log* do sistema, que se encontram dentro da diretoria */var*. É ainda importante analisar a directoria */tmp* de cada utilizador do sistema.

2.2.2 Sistemas *Windows*

O *Windows* é o SO mais popular devido a vários fatores, sendo o mais importante a sua grande facilidade de utilização e a forma como foi difundido pelo mundo empresarial e domestico. Na imagem 2.2 podemos ver que atualmente tem uma cota de mercado de cerca de 90%, neste valor estão incluídas as suas várias versões [11].

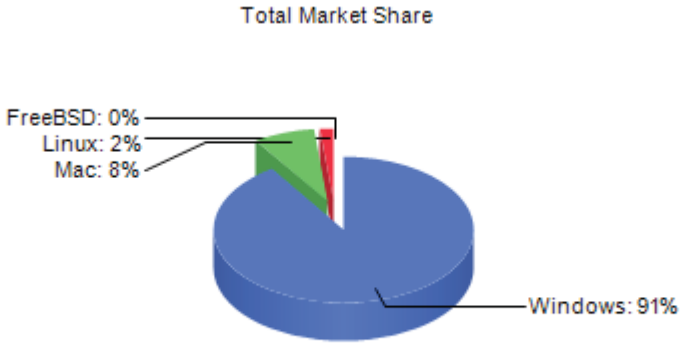


Figura 2.2: Utilização dos SO

O processo de Análise Forense para o SO *Windows* consiste na recolha de evidências que se julguem importantes. Essas evidências estão geralmente presentes em logs de eventos, *Slack Space* (o espaço do *cluster* que não foi totalmente ocupado), o registo de *Windows* e ficheiros temporários, ficheiros apagados, dados em memória. Os ficheiros que estão relacionados com ambiente de funcionamento do *Windows* é de extrema importância a sua recolha. Os *logs* têm informações sobre dados relacionados com eventos que podem não afetar o sistema, por exemplo a mudança de utilizador, as alterações de permissões, *logon/Logoff*. O registo do *windows* contêm informações sobre acessos a ficheiro, pastas do utilizador, entre outros. E importante ter em conta que existem mais "lugares" dentro do SO onde se podem recolher evidências, mas estes são os pontos mais importantes.

***Logs* de eventos** A informação contida nos *logs* de eventos é uma fonte de dados importantes, principalmente quando se trata de recolha de provas relacionadas com ataques intrusivos com fins maliciosos. O *log* de eventos do *Windows* é a fonte mais

importante de provas durante a investigação forense de um sistema *Windows*, pois os ficheiros de *log* permitem conectar certos eventos com um determinado intervalo temporal [12].

A arquitetura do SO *Windows* tem medidas de segurança adequadas para o registo dos logs, direcionando o registo de eventos com base no seu tipo e que ocorrem no SO, facilitando assim o registo e a consulta. Os logs são gerados quando o SO inicia, numa falha de funcionamento, quando um utilizador tenta aceder a recursos do sistema ou se liga a um computador, rede, *Internet*, controlo de processos, eventos do sistema, entre outros. Estes vão gerando em detalhe todas atividades que ocorrem no SO O *Windows* tem um sistema de registo de segurança, que só pode ser escrito pelo LSASS (*Security Authority Subsystem Service Local*). A política de segurança do SO *Windows* é implementada pelo processo LSASS sendo uma parte vital da segurança do SO entre as várias funcionalidades que são fornecidas pelo LSASS temos, por exemplo, a verificação de utilizadores no *Login* do *Windows*, manipulação de *password*, criação de tokens de acesso e, consequentemente, escrever entradas no log de segurança do *Windows*.[12]

Windows Registry O Registo do *Windows* contém informações importantes sobre o *software* instalado no computador, mas também mantém informações sobre as atividades do utilizador. Algumas das chaves mais importantes que estão no registo são:

- Lista MRU (*Most Recently Used*): é uma lista que contêm as acções mais recentes executadas pelo utilizador. Todos os comandos executados criam uma nova entrada na chave de registo. Essencialmente, o seu funcionamento é semelhante à forma como o histórico e *cookies* do *browser web* funcionam;
- *Startup Objects*: São os objetos que estão definidos para iniciar automaticamente quando o SO inicia;
- *Internet Explorer*: guarda os seus dados numa chave com três subchaves inseridas na primeira, e que detém a maioria das informações úteis;
- *UserAssist*: Esta chave contém duas ou mais subchaves, que têm nomes hexadecimais ou identificadores globais exclusivos (GUIDs) e abaixo de cada GUID é uma subchave chamada Contagem. A subchave contém valores registrados que dizem respeito a objetos que o utilizador acedeu, tais como applets, painel de controlo, ficheiro de atalhos, programas, documentos, ente outros;

- USB *Devices*: Sempre que um dispositivo é ligado ao (USB) *timeUniversal Serial Bus*, os *drivers* são verificados e as informações sobre o dispositivo são armazenados no registo (ou seja, *pen drives*, câmeras, entre outros.) A chave contém subchaves que representam a identificação de qualquer dispositivo USB que foi ligado ao sistema;

- *Computers Network*: O SO tem uma ferramenta de gestão de Rede, *My Network Place*, que facilita as tarefas relacionadas com a gestão da Rede. O registo contém todos os computadores de uma mesma rede, mesmo depois do computador ter sido desligado esse computador permanece no registo [13].

2.2.3 Informação Volátil

Obter dados com o equipamento em funcionamento facilita muito o trabalho de um investigador, para além da vantagem de obter um conjunto de dados, que de outra forma seria muito complicado ou mesmo impossível. Os principais dados que se conseguem obter são as ligações de rede e os dados existentes em memória do equipamento. Informações como processo em execução, serviços ativos, informações do sistema, *password*s, informação que não tenha sido guardada, ligações de rede abertas, utilizadores ligados e conectados ao equipamentos, informações de registo, aplicações com dados cifrados, *malware* existente, ARP (*Address Resolution Protocol*)*cache*, indicação de utilização de técnicas anti-forenses, *backdoor* ativas, entre outros.

Antes de se realizar uma análise deverá ser feita uma avaliação do risco, conforme as diretrizes normais. Deverá avaliar-se se é seguro e proporcional a captura dos dados em tempo real e em que modos isso poderá influenciar positivamente a investigação, a metodologia utilizada tem de ter como principal objetivo a forma como captura os dados e preserva toda a informação obtida.

Em resumo os passos a serem executadas devem ser os seguintes:

- Realizar uma avaliação de risco da situação;
- Instalar dispositivo de captura de dados voláteis;
- Execute o *script* de captura de dados voláteis;
- Uma vez concluída, interrompa o dispositivo;
- Remova o dispositivo;
- Verifique se a recolha de dados foi correta numa máquina de investigação.

Uma situação que representa um desafio para o investigador, encontra-se no sistema informático num ambiente corporativo, devido às restrições que a instituição pode ter. A situação estará facilitada se existir na rede *software* de análise forense que consiga criar uma imagem da rede. Outros equipamentos que podem ajudar na investigação são os *router* e *firewalls* que fornecem informação sobre a configuração da rede. No caso de uma rede de grandes dimensões é importante o aconselhamento e assistência do administrador de rede, assumindo que não existem suspeitas sobre eles [14].

2.2.4 Ferramentas Forenses

Existem dois tipos básicos de dados que são importantes, os dados em disco e os dados voláteis. Existe uma variedade de ferramentas utilizadas para recolher dados. Sendo difícil encontrar um que responda a todos os requisitos necessários para uma correta recolha de informação. Dai existirem os *kits* de *softwares* forenses, que desempenham diversas funções integradas como *backup*, autenticação, criptografia, edição de disco, *log* de auditoria de ficheiros, monitorização de IP (*Internet Protocol*), recuperação de dados e analisador de ficheiros.

Uma ferramenta de captura de dados fiável deve estar de acordo com as exigências do NIST (*National Institute of Standards and Technology*)

- A ferramenta deve duplicar um fluxo de *bits* ou uma imagem do disco original ou seção;
- A ferramenta não deve alterar o disco original, ou seja, o programa não pode fazer alterações às evidências originais;
- A ferramenta deve ser capaz de verificar a integridade de imagem dos dados;
- Erro de I/O a ferramenta deverá fazer o *log* de actividade, ou seja, este programa deve oferecer uma solução para corrigir erros;
- A saída da documentação registrada deve ser correcta [15].

O *Software* de autenticação é utilizado para certificar que a prova não foi alterada. Os programas utilizam os algoritmos MD5 ou o SHA-1 (*Secure Hashing Algorithm 1*) para gerar valores de *hash*. São produzidos *hash* para os dados originais e para as cópias de forma a verificar que são idênticas, confirmando assim, a sua autenticidade.

As Ferramentas de Descodificação são necessárias para obter acesso ao computadores, ficheiros protegidos por *password*. Se os dados recuperados do computador estiverem cifrados deverão ser aplicados métodos para tentar determinar a cifra e obter os dados. Nomeadamente *screensavers*, documentos *Office*, PDF, ficheiros comprimidos, em que para todos existem vários programas que podem ser utilizados quando estão protegidos por medidas de segurança, como *password*s.

Quando os dados são apagados pelo utilizador eles ainda permanecem no disco rígido. Não estado visíveis nem com indicação do espaço que ocupam no disco. Existe um código de identificação que está ligado ao título do documento que representa o facto de o documento ter sido apagado. Para recuperar os ficheiros apagados utiliza-se uma ferramenta de recuperação de dados que esta configurada para pesquisar por este tipo de identificação, analisando e verificando se é possível recuperar esses ficheiros. Os documentos recuperados são de vários formatos, dai ser necessário ter um *software* para se proceder à sua correta leitura, o ideal será ter um *software* que consiga ler diferentes formatos de ficheiro.

Realizar análises aos ficheiros de *logs* é um procedimento importante, visto conterem registos de atividades no SO, como por exemplo o ficheiro de *log* do *browser*. Os *Logs* são um instrumento importante juntamente com algumas ferramentas que ajudam a obter informações adicionais sobre os endereços IP, como o comando *ping* ou *traceroute* para SO *Windows*.

São muitas as ferramentas que se podem utilizar para facilitar o processo de recolha de provas num dispositivo, existem muitos *softwares* que se encaixam nestas categorias, a melhor opção será escolher o *software* que se adapte melhor à situação em análise e que venha a ter uma hipótese muito reduzida de comprometer as provas recolhidas [16].

2.3 Novos Desafios

2.3.1 *Cloud Forensic*

A *Cloud Computing* tem-se tornado numa das tecnologias informáticas mais emergentes e transformadoras dos últimos anos, com uma capacidade de integração muito grande desde computadores, servidores, *smartphones*, entre outros.

Cloud Computing é um modelo para permitir um acesso continuo através da

rede a um conjunto de recursos computacionais partilhados e configuráveis, como por exemplo redes, servidores, aplicações e serviços que podem ser rapidamente fornecidos com um esforço mínimo de gestão ou interação com o fornecedor de serviços. O modelo de *cloud* é composto por cinco características essenciais, três modelos de serviço e quatro modelos de implantação [17].

Tabela 2.4: Modelo *Cloud*

Características	**Modelos de serviço**	**Modelos de implantação**
On-demand self-service	*Software as a Service (SaaS)*	*Private Cloud*
Broad network access	*Platform as a Service (PaaS)*	*Community Cloud*
Resource pooling	*Infrastructure as a Service (IaaS)*	*Public cloud*
Rapid elasticity		*Hybrid cloud*
Measured service		

A *Cloud Computing* está a alterar radicalmente a forma como os serviços de tecnologia estão a ser criados, acedido e geridos. Assim como os serviços estão a crescer também o volume de informação aumenta, criando assim, um novo problema para a Computação Forense, visto que o modelo de informação da *Cloud Computing* cria novos crimes cibernéticos que por sua vez geram novos desafios à investigação forense.

Cloud Forensic

Cloud Forensic é um subconjunto da *Network Forensics*. A *Network Forensic* lida com investigações forenses de redes informáticas. Portanto, *Cloud Forensic* segue as principais fases da investigação da *Network* Forenics com técnicas adaptadas para investigações em sistemas *Cloud Computing*.

Dois dos maiores problemas que a *Cloud Forensic* enfrenta, são o volume de informação e a forma de aceder aos dados. O enorme volume de informação implica uma grande capacidade para analisar os dados em tempo útil e a localização dos mesmos, que podem estar alojados e replicados em vários locais, sem que o acesso pela entidade gestora seja facultado.

Dimensão Técnica

A questão técnica tem em conta os processos e ferramentas que são necessárias para realizar o processo de computação forense na *Cloud Computing*, como a recolha de dados, análise forense ao sistema em funcionamento, identificação das provas,

sistemas virtualizados e medidas pró-ativas.

A recolha de dados na *Cloud* apresenta dificuldades, devido ao facto de estarem num sistema complexo e de poderem ser feitas em dois locais, no cliente e no fornecedor de serviços. Os procedimentos e ferramentas a utilizar vão depender muito do modelo que o fornecedor tiver implementado, mas sempre com o cuidado de preservar a integridade dos dados e de acordo com os procedimentos jurídicos em vigor.

A capacidade de se adaptar as necessidades é uma das vantagens da *Cloud* Compuntig, o que implica ter ferramentas que consigam perceber as alterações que ocorrem no sistema e registem esses acontecimentos. Sendo um ambiente de partilha de recursos, é preciso ter procedimentos para conseguir fazer uma análise só aos dados definidos.

A virtualização é a tecnologia chave que é utilizada para implementar os serviços da *Cloud*. O maior problema que a investigação tem são os procedimentos de uma investigação num ambiente virtualizado, por serem poucos, e o controlo dos dados. Sendo necessário definir procedimentos para conseguir localizar informação com data e hora especificas, tendo em conta as questões jurídicas. A melhor solução será aplicar medidas que facilitem a investigação forense, como preservar imagens dos dados, uma monitorização contínua de autenticação, controlo de acesso e dos acessos ao nível dos objetos.

Dimensão Organizacional

No ambiente da *Cloud* são pelo menos duas as entidades envolvidas na investigação forense, o cliente e o fornecedor de serviços, nos casos em que o fornecedor descentralizar os seus serviços, o numero de entidades pode aumentar. O que vem a dificultar o processo, por criar uma cadeia de investigação maior e com o risco de falha de comunicação entre as partes.

Dimensão Legal

Sendo a *Cloud Computing* e as suas respetivas tecnologias, consideradas novas em relação a legislação vigente, a ausência de normas reguladoras e específicas que criminalizam ilícitos virtuais torna a necessidade de utilizar as leis existentes para solucionar possíveis implicações jurídicas.

É importante definir os contratos de fornecimentos de serviços, especificando de forma clara as responsabilidades de cada uma das partes. Exigindo garantia por parte dos fornecedores de que os seus sistema de segurança atendam às questões

legais, em conformidade com as leis locais em vigor, uma vez que os dados da nuvem podem estar alojados em um ou vários países, onde as leis em vigor podem ser contrárias às leis do pais. Desta forma a dimensão jurídica da *Cloud Computing* requer a implementação de regulamentos e acordos de forma a garantir que o desenvolvimento das atividades forenses não violam as leis e regulamentos onde os dados estão alojados.

2.3.2 Dispositivos Móveis

Os dispositivos móveis transformaram a forma de comunicar. Um avanço na tecnologia que colocou a capacidade de processamento de um computador e a capacidade de comunicar de um telefone num dispositivo de dimensões reduzidas. O primeiro passo foi dado com a criação do PDA (*Personal digital assistant*), que foi construido para os utilizadores acederam à *web*, *email*, entre outras funções. A partir dai com os avanços tecnológicos tanto ao nível de *hardware* como de *software* resultaram nos atuais dispositivos preparados para serem plataformas de trabalho e diversão móveis, com variadíssimas aplicações desde jogos a ferramentas de trabalho.

Um dos equipamentos que mais impulso deu foi o *iPhone*, da *Apple*, caracterizado por ser um equipamento com diversas aplicações que estende as capacidades dos dispositivos muito para além da comunicação móvel. A conjugação dos dispositivos com *hardware* e *software* da *Apple*, resultaram num dos melhor dispositivos que existem. Numa visão diferente surge a *Google*, com o SO *Android* que domina o mercado pela diversidade, que tem um SO, com diversas versões e instalado por diferentes fabricantes, em dispositivos de vários tipos, de baixo custo a topo de gama.

Mas estes avanços não se deram somente pelas questões tecnológicas dos dispositivos móveis, os avanços na tecnológicas de comunicação vieram conjugar todos esses avanços, desde o aumento da largura de banda à diminuição do custo de utilização, sendo fortes impulsionadores do aumento da mobilidade do utilizador, o que levou a uma liberdade de comunicação sem fronteiras, permitindo aos utilizadores estarem continuamente ligados à *web*.

O paradigma da comunicação foi completamente alterado, com a junção de vários fatores que criam um novo mercado que rapidamente foi aproveitado pelas empresas, neste momento os dispositivos móveis têm uma amplitude de utilização muito grande, em várias áreas de negócio e de diversão.

Os recursos dos dispositivos móveis estão em constante mudança, por isso é

difícil definir o termo "dispositivo móvel". No entanto, é importante estabelecer uma base de recursos de dispositivos móveis. As seguintes características definem os dispositivos:

- Dimensões reduzidas;
- Pelo menos uma interface de rede sem fios para o acesso à rede (comunicações de dados). Esta interface usa *Wi-Fi*, redes telefónicas ou outras tecnologias que conectam o dispositivo móvel a infraestruturas de rede com conectividade com a Internet ou outras redes de dados;
- Armazenamento de dados;
- Aplicações.

A lista a seguir apresenta outras características comuns, mas opcionais, de dispositivos móveis. Esses recursos não definem a base dos dispositivos mas indicam características que são particularmente importantes em termos de segurança.

- Serviços de rede;
  - + Uma ou mais interfaces de rede *wireless*;
  - + Uma ou mais interfaces de rede sem fio para comunicações de voz;
  - + GPS (*Global Positioning System*), que permite serviços de localização;
- uma ou mais câmaras digitais/gravação de vídeo;
- armazenamento;
- Suporte para utilizar o próprio dispositivo como armazenamento removível para outro dispositivo de computação;
- Recurso para sincronização de dados [18].

Plataformas utilizadas

Nos Dispositivos Móveis existe uma grande variedade de SO, com diferentes versões, por vezes adaptadas pelos fabricantes especificamente para cada dispositivo.

O *Android* é o SO criado pela *Google*. Com um modelo *Open Source*, assim, acessível a todos os interessados. Baseado no núcleo *Linux*, suporta qualquer tipo de conexão sem fio 3G, EDGE (*Enhanced Data rates for GSM Evolution*), *Wi-Fi* e *bluetooth*. Sendo compatível com a maior parte dos ficheiros de multimédia. Como é

Open Source ele é capaz de ser modificado, para se adaptar a equipamentos de baixo custo. Conta já com inúmeras aplicações para personalizar o equipamento [19].

O IOS é o SO para os dispositivos *iPhones* e *iPads*, é um sistema derivado do *Mac OSX*. Este foi o primeiro SO criado para dispositivos móveis que tem suporte para as tecnologias de toque múltiplas, aptas para ficheiros multimédia, para navegar na *web*, trabalhar com aplicação ou telefonar, para isso basta arrastar o dedo pela interface gráfica e para abrir uma aplicação basta dar um pequeno toque sobre o ícone da aplicação. O lado menos interessante desse sistema é que as suas aplicações só estão disponíveis nas lojas da *Apple* [19].

O *Windows Mobile* é um SO para *smartphone* baseado no *Kernel* do *Windows CE6*. Utilizado nos *Pocket Pcs*. Utiliza o mesmo padrão de aplicações que a versão para computador, mas requer um *hardware* com boa capacidade de processamento para funcionar corretamente. Sendo compatível com todas as aplicações básicas para a versão PC: *Word*, *Excel*, *Power Point*, *Windows Media Player*, entre outros [19].

O *BlackBerry* é um SO concebido pala empresa RIM (*Research in Motion*). Integra diversas funções importantes e que foram pela primeira vez utilizadas em Smartphone, editor de texto, acesso à *internet*, *email* e tecnologia IPv6. O que o diferencia dos demais, é que o *BlackBerry* utiliza um serviço próprio de *email* chamado BBM (Balckberrrymessenger). As mensagens de *email* no envio e receção chegam até 200*kbps*, utilizando a tecnologia EDGE [19].

Utilização

Os *smartphones* são dos dispositivos móveis mais vendidos no mundo das comunicações móveis, sendo a maioria baseada em *Android*, seguido do iOS, *Windows Phone*, *BlackBerry* e assim por diante. Foram responsáveis por 250 milhões de aparelhos, ou 55% do total. Em comparação, a 171 milhões de vendas de *smartphones* no ano de 2012 que ficou com uma quota de pouco menos de 40 % do total. Tem-se verificado um aumento gradual todos os anos, neste momento a venda de *smartphones* ultrapassou a venda de telemóveis.

Na luta pelas vendas a *Samsung* lidera o mercado global, agora com uma quota de mercado de 32%, seguida da *Apple*, com seu *smartphone iPhone*, com uma percentagem de 12,1% de vendas o que é equivalente a 30 milhões de unidades.

Outra marca que é importante referir é a *Lenovo*, que colocou a LG em terceiro lugar com 12,9 milhões de *smartphones* vendidos. O sucesso da *Lenovo* deve-se as suas vendas no mercado Chinês.

Worldwide Smartphone Sales to End Users by Operating System in 3Q13 (Thousands of Units)

Operating System	3Q13 Units	3Q13 Market Share (%)	3Q12 Units	3Q12 Market Share (%)
Android	205,022.7	81.9	124,552.3	72.6
iOS	30,330.0	12.1	24,620.3	14.3
Microsoft	8,912.3	3.6	3,993.6	2.3
BlackBerry	4,400.7	1.8	8,946.8	5.2
Bada	633.3	0.3	4,454.7	2.6
Symbian	457.5	0.2	4,401.3	2.6
Others	475.2	0.2	683.7	0.4
Total	**250,231.7**	**100.0**	**171,652.7**	**100.0**

Source: Gartner (November 2013)

Figura 2.3: Marcas de dispositivos móveis

O *Android* mantém a liderança nos SO, com 81,9% resultado em mais de 200 milhões de instalações, com uma grande vantagem sobre o iOS que fica em segundo lugar. A plataforma móvel da *Apple* ficou em segundo lugar com 12%. Apesar da diferença entre o *Android* e iOS, o sistema operativo do *smartphone* da *Apple* conseguiu terminar confortavelmente à frente do *Windows* Phone da *Microsoft*. A Plataforma de *smartphone* da *Microsoft* teve um aumento em 2013 mas são valores bastante reduzidos, não sendo significativo. Os restantes SO têm pouco impacto no mercado, o que demonstra que neste momento há poucas oportunidades para novos sistemas, que possam surgir.

Neste momento o mercado dos SO tem um claro vencedor o *Android*, devido a vários fatores, onde saliento o custo de instalação e a portabilidade.

Worldwide Smartphone Sales to End Users by Operating System in 3Q13 (Thousands of Units)

Operating System	3Q13 Units	3Q13 Market Share (%)	3Q12 Units	3Q12 Market Share (%)
Android	205,022.7	81.9	124,552.3	72.6
iOS	30,330.0	12.1	24,620.3	14.3
Microsoft	8,912.3	3.6	3,993.6	2.3
BlackBerry	4,400.7	1.8	8,946.8	5.2
Bada	633.3	0.3	4,454.7	2.6
Symbian	457.5	0.2	4,401.3	2.6
Others	475.2	0.2	683.7	0.4
Total	**250,231.7**	**100.0**	**171,652.7**	**100.0**

Source: Gartner (November 2013)

Figura 2.4: SO para dispositivos móveis

As dificuldades de se fazer uma análise forense

Os dispositivos móveis são sistemas com um grande dinamismo o que representa grandes desafios para os investigadores forenses. A diversidade de modelos que existem juntamente com as suas constantes evoluções e a sua crescente introdução no mercado das telecomunicações torna difícil desenvolver um único procedimento ou

ferramentas que sejam capazes de realizar uma análise a todos os pontos importantes.

Ao nível dos SO existem várias versões por vezes adaptadas para cada dispositivo, pelas empresas que fabricam os equipamentos. Existindo assim, um grande problema nos equipamentos de baixo custo, devido às alterações que são introduzidas nos SO para se adaptarem aos dispositivos, o que dificulta a definição de padrões de análise para os SO. Quando preservamos evidências nos dispositivos móveis e devido às suas características, existem considerações importantes a ter em conta. A maioria dos dispositivos móveis são equipamentos de rede, permitindo o envio e receção de dados através de sistemas de telecomunicações, o acesso *Wi-Fi e Bluetooth*, as evidências digitais nos dispositivos móveis podem ser completamente perdidas devido a poderem ser substituídas por novos dados que são recebidos ou enviados. Além disso, é preciso ter em conta que é necessário interagir com o dispositivo, o que pode alterar o estado do sistema e por consequência alterar ou destruir dados existentes.

Os dispositivos móveis são um desafio para os analistas, desde a recuperação de dados até a sua análise. As suas funcionalidades e capacidade de armazenamento de dados têm tido um crescimento elevado, chegando ao ponto de rivalizarem em determinados aspetos, como disponibilidade de acesso a dados na *web*.

No entanto ao nível da recuperação de dados os dispositivos móveis têm uma vantagem sendo a principal razão os *chips* de memória *flash*, que são extremamente duradouros e pela forma como armazenam os dados. A forma como é feita a eliminação dos dados facilita a recuperação, os dados só podem ser apagados bloco a bloco, geralmente o bloco só é apagado quando esta cheio, sendo assim, fica muita informação preservada em memória. Uma das vantagens poucas vezes referidas, consiste na utilização pessoal do dispositivo, o que faz com que seja possível estabelecer uma linha de ação do utilizador, facilitando a caracterização dos acontecimento para o analista.

Procedimentos Generalistas

Os procedimentos de investigação podem ser vários, consoante as situações apresentadas. Numa das publicações do NIST [20] é possível encontrar os procedimentos generalistas para o processo de investigação, sendo este documento um dos primeiro a definir esses procedimentos. Fornecendo informações sobre a preservação, aquisição, análise e documentação de evidências digitais em dispositivos móveis. Focando-se em detalhes e características específicas dos equipamentos, salientando que cada in-

vestigação tem as suas especificidades únicas.

Noutro guia de procedimentos pode-se encontrar numa publicação da ACOP *(Association of Chief Police Officers)*, uma serie de detalhes e princípios para garantir as boas práticas da recolha de evidências. São feitas referências as ações em cenas de crime, preservação, documentação, análise inicial e aos diversos tipos de equipamentos que existem no mercado [14].

2.4 Modelo de Computação Forense

O processo de investigação forense é descrito por Casey [21] como uma sequência de processos ascendentes, sendo um modelo que permite descrever metodicamente todas as etapas de uma investigação de forma estruturada, rigorosa e completa. Na imagem 2.5, as etapas estão descritas de uma forma genérica, onde se pretende incluir as funções de âmbito policial e as tarefas dos peritos forenses, sendo que o modelo aplica-se vários tipos de investigações, mesmo as militares e as aplicadas feitas num ambiente empresarial.

Em geral, este modelo proporciona aos investigadores e examinadores um fluxo lógico de eventos que juntos, pretende oferecer:

1. Aceitação - as etapas e métodos ganharam consenso profissional;
2. Confiabilidade - os métodos são utilizados para gerar os resultados;
3. Repetição - o processo pode ser repetido, independente de tempo e lugar;
4. Integridade - evidência não é alterada;
5. Causa e efeito - contexto lógico entre indivíduos suspeitos, eventos e exposições;
6. Documentação - documentos para servirem de prova.

Todos os seis princípios têm um propósito comum, formar o argumento mais persuasivo é válido possível, com base em fatos e não em suposições e fazê-lo considerando os critérios legais de admissibilidade.

Alerta de incidente ou Acusação – o alerta de incidente ou acusação é o início de todo o processo. Dentro desta fase, em primeiro lugar, as fontes são avaliadas e informações são solicitadas.

Figura 2.5: Modelo Forense

Avaliação do valor – nesta fase é efetuada uma avaliação dos custos de prosseguir com a investigação. As desvantagens de um processo residem na necessidade de recursos, possível tempo de inatividade do sistema investigado e por vezes o efeito negativo da investigação para a organização. Por regra toda a suspeita de atividade criminosa deve ser investigada, por vezes como melhoria do sistema de segurança e como efeito de dissuasão.

Protocolo de procedimento – no procedimento de um criminalista clássico, toda a cena do crime é fechada para investigação. Para os incidentes digitais o processo é semelhante, os diferentes tipos de vestígios digitais têm de ser verificados individualmente com base num processo de documentação, o estado e que garanta a integridade dos mesmos com um risco mínimo de serem alterados. É importante referir que as informações obtidas durante esta etapa sobre o estado do incidente contêm os elementos que são analisados a um nível mais elevado ou seja com um nível de detalhe digital mínimo.

Identificação ou apreensão – Neste ponto é necessário compreender os procedimentos e critérios legais necessários, para que se possa tomar decisões de forma a aproveitar tudo o que esteja envolvido na cena (físico ou virtual) de forma a não provocar alterações às provas e estar preparado para documentar e justificar ação. Documentar o processo de investigação é fundamental, mas é particularmente importante na etapa de apreensão de evidência digital. É necessário registrar os detalhes sobre cada pedaço de provas apreendidas para ajudar a estabelecer a sua autenticidade e iniciar a cadeia de custódia.

Como base de procedimentos temos o guia do *US Department of Justice*, *"Electronic Crime Scene Investigation: A Guide to First Responders"* [22] e o documento *"Searching and Seizing Computers and Obtaining Electronic Evidence in Criminal Investigations"* [23] vocacionado para pessoal não técnico. O guia publicado pelo ACOP "Good Practices Guide for Computer Based Electronic Evidence" [14], fornece um ponto de partida estabelecendo quatro princípios gerais para diversos dispositivos electrónicos, nomeadamente.

1. Nenhuma acção tomada pela polícia ou pelos seus agentes, devem alterar os dados armazenados num computador ou noutros meios de comunicação que podem, posteriormente, ser invocado em tribunal.

2. Em circunstâncias excepcionais, onde se ache que é necessário aceder os dados originais armazenados num computador, então essa pessoa deve ser competente para o fazer e para prestar depoimento explicando a relevância e as implicações de suas acções.

3. Todo o procedimento de auditoria ou de registro aplicados a evidência baseada em computador deve ser criado e preservado. Para que todo o procedimento possa ser possível ser reproduzido e nele se obter o mesmo resultado.

4. O oficial encarregado do caso é responsável por garantir que a lei e estes princípios são respeitados. Isto aplica-se à posse e acesso a informações contidas num computador.

Preservação – Todas as evidências recolhidas têm de estar certificadas de forma a que os seus itens permaneçam inalterados. Para as evidências digitais, significa que em primeiro lugar, são criadas cópias das provas e a investigação mais aprofundada é feito apenas nas cópias. Para provar a autenticidade da cópia da prova, são utilizadas ferramentas criptográfico para fazer essa confirmação através de funções

hash. Durante esta fase é que se inicia o trabalho dos peritos forenses.

Recuperação – Antes de realizar uma análise completa das fontes de evidências digitais, é necessário para extrair os dados que tenham sido apagados, escondido, ou que estão de outra maneira disponível para visualização utilizando o sistema operativo nativo e do sistema de ficheiros existentes. Nesta etapa do processo o foco é sobre a recuperação de todos os dados não disponíveis ou que sejam pertinentes para o caso ou incidente. O objetivo é identificar, e se possível tornar visível, todos os dados que possam ser reconhecidos como pertencentes a um determinado tipo de dados. Fornecendo o máximo de informação para as próximas fases do processo.

Recolha – Durante a análise das provas devem-se organizar de forma estruturada devido à enorme quantidade de dados. Por essa razão, para facilitar o processo investiga-se primeiro os metadados em vez dos dados reais. Os dados podem ser agrupados de acordo com o tipo de ficheiro ou tempo de acesso, conforme a finalidade da investigação.

Redução – A tarefa de redução centra-se na eliminação de dados irrelevantes. Dai a importância dos Metadados e de uma boa estruturação dos dados.

Organização e Pesquisa – Os aspetos da organização são a estruturação de dados por dados significativos, bem como a pesquisa pelos mesmos. Por isso, muitas vezes, índices e resumos são criados ou ficheiros são classificados por tipo de informação. Isto simplifica o referenciamento de dados para as próximas etapas. O objetivo principal desta atividade é fazer com que seja mais fácil para o investigador encontrar e identificar os dados durante a etapa de análise e permitir-lhes fazer referência a esse dado de uma forma significativa nos relatórios. Esta atividade pode incorporar diferentes níveis de tecnologia de busca para ajudar os investigadores a localizar potenciais evidências. As ferramentas ou tecnologia utilizada nesse sentido, devem seguir os padrões aceites, para que os resultados desta etapa se possam repetir.

Análise – Esta etapa envolve a análise detalhada dos dados identificados nas atividades anteriores. As técnicas empregadas aqui tendem a envolver análise e estudo de atributos específicos, internos dos dados ou o formato específico de itens de áudio e dados de vídeo binários. Além disso, características de classe e individuais encontrados nesta etapa são utilizados para estabelecer ligações, determinar a origem de itens, e, finalmente identificar o infrator. Geralmente, a análise inclui as seguintes

subcategorias

- Avaliação - legível (ou visíveis) objetos de dados digitais humanos têm conteúdo ou substância que pode ser percebido. Essa substância será examinada para tentar determinar fatores como meio, motivação, oportunidade.
- Experimentação - Um termo muito geral, mas aplicado aqui para dizer que os métodos e técnicas não ortodoxas ou previamente inexperiente podem ser utilizados durante as investigações. Todas as metodologias comprovadas como experiências têm de ser rigorosamente documentadas, especialmente quando se aplicam procedimentos científicos.
- Fusão e correlação - Durante o curso da investigação, os dados (informações) foram recolhidos de várias fontes (digitais e não-digitais). As evidências digitais por si só não vão apresentar o relato total do incidente. O inverso também é verdadeiro. Os dados devem ser fundidos ou reunidos para preencher as estruturas necessárias para contar a história completa. Um exemplo de fusão seria o cronograma de eventos associado a um caso ou incidente em particular. Cada crime ou incidente tem uma componente cronológica em que eventos ou ações vão preencher fatias de tempo. Correlação é o relacionamento, mas tem mais a ver com a causa e efeito fundamentado.
- Validação – E o resultado da fase de análise. São as conclusões fundamentadas que os investigadores propõem a submeter-se a juristas ou outros tomadores de decisão como "uma prova positiva"para a acusação ou absolvição.

são utilizados para estabelecer ligações, determinar a origem de itens, e, finalmente identificar o infrator. Geralmente, a análise inclui as seguintes subcategorias

Relatórios – O relatório não é apenas para a apresentação dos resultados, mas também para demonstrar como se chegaram aos resultados obtidos. Para isso, todas as regras e padrões considerados devem ser documentadas. Além disso, todas as conclusões devem ser justificadas e modelos explicativos.

Persuasão e Testemunho – Trata-se do testemunho como uma autoridade sobre o assunto em Tribunal. O aspecto mais importante é a confiabilidade da autoridade. Um público adverso a questões tecnológicas pode ser problemático.

Capítulo 3

Computação Forense em Sistema *Android*

3.1 Sistema *Android*

O *Android* é uma plataforma de código aberto para dispositivos móveis essencialmente baseada no *Kernel* 2.6 do *Linux*, sob a licença *Apache* 2.0 de 2004 e mantida atualmente pela OHA *Open Handset Alliance*, um grupo de diversas empresas de diferentes ramos industriais como operadoras de comunicações, fabricantes de telemóveis, fabricantes de componentes e desenvolvimento de softwares, entre outras.

"Android was built from the ground up with the explicit goal to be the first open, complete, and free platform created specifically for mobile devices." OHA [24].

O objetivo principal do *Android* é de modificar experiências na utilização de dispositivos móvel pelos consumidores, deixando-a melhor, mais rica e menos dispendiosa, com o compromisso de inovar e desenvolver os dispositivos *Android* [24].

A questão de ser uma plataforma livre e de código aberto apresenta vantagens para as empresas, programadores e consumidores, desde o facto de poder ser utilizado gratuitamente, reduzindo os custos, e por poder ser personalizado, dando liberdade às empresas para criarem adaptações para os seus dispositivos.

A *Google Inc.*, tomou a decisão de utilizar a licença de *Apache* 2.0 por ter grande aceitação comercial, já que é menos restrita e não força as empresas a abrir o código fonte de todo o seu *software*, personalizado ou não [25].

A plataforma *Android* é composta basicamente pelo sistema operativo, o SDK (*Software Developmemt Kit*) e as aplicações. O SDK é de fácil acesso e utilização

tendo como linguagem de desenvolvimento padrão o JAVA (*Just Another Virtual Architecture*), posteriormente tendo sido tomada a decisão de não utilizar a plataforma padrão do JAVA e utilizar uma máquina virtual DVM (*Dalvik Virtual Machine)* permitindo uma utilização das aplicações em todos os equipamentos.

A Arquitetura do *Android* é formada por camadas que intercomunicam mas dependentes entre si, onde as camadas de baixo nível fornecem serviços para as camadas de nível superior. O seguinte diagrama da arquitetura dá uma visão mais ampla e fornece um entendimento melhor sobre a estrutura da plataforma.

Segundo o modelo, existem quatro camadas com cinco grupos diferentes, o *kernel Linux* que fornece a interface de baixo nível com o *hardware*, as bibliotecas open source para desenvolvimento de aplicações, um ambiente de execução (máquina virtual), que executa e guarda aplicações, a *framework* de aplicações que fornecem serviços do sistema para a camada de aplicação e a própria camada de aplicação, onde estão as aplicações.

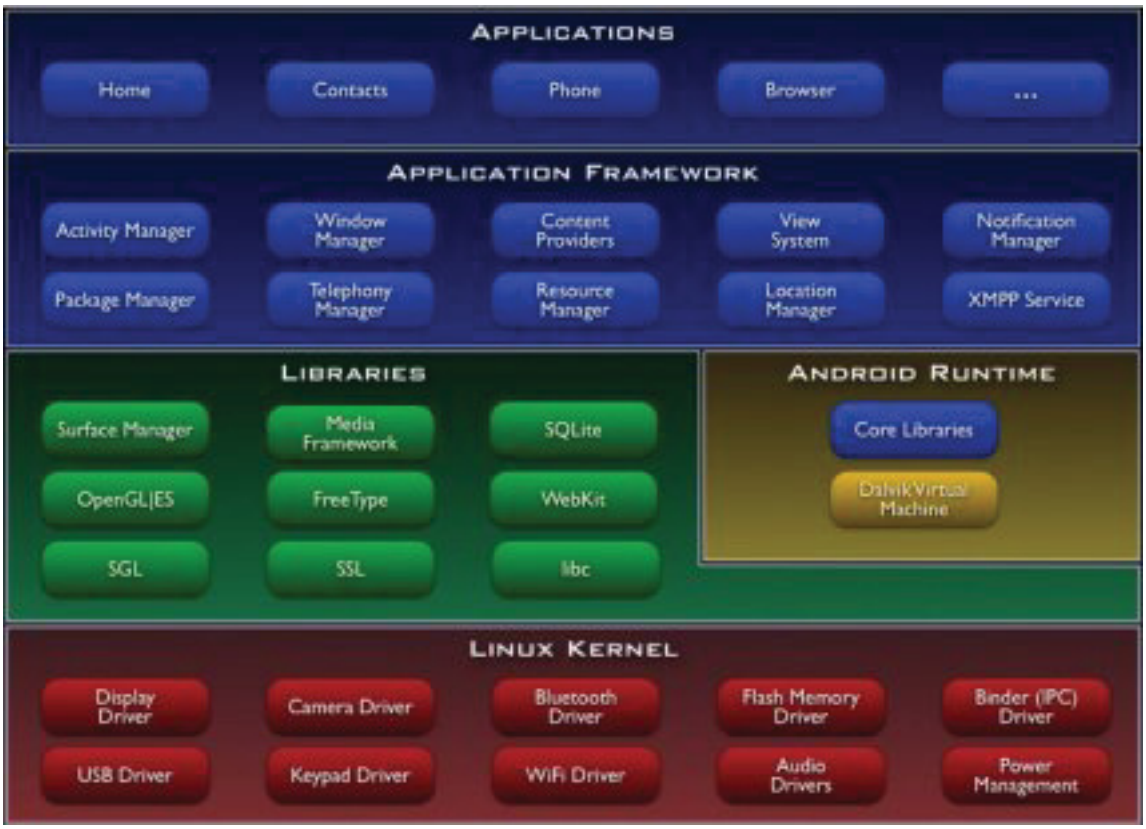


Figura 3.1: Arquitetura *Android*

Camada de Aplicações: a plataforma *Android* vem com um conjunto básico

de aplicações, *browser*, *email*, SMS (*Short Message Service*), contactos, entre outros. Todo o sistema foi pensado para ser multitarefa, o que permite ao utilizador realizar várias tarefas em simultâneo, as aplicações são desenvolvidas em JAVA. [24]

***Framework* aplicacional:** é uma *framework* de desenvolvimento padronizado e aberto que permite com a ajuda de fornecedores de conteúdos e outros serviços a reutilização de funções das aplicações e dos seus recursos. Todas as API (*Application Programming Interface*) disponíveis para o sistema principal também estão disponíveis para o desenvolvimento das aplicações, permitindo aos programadores terem todos os recursos disponíveis. [24]

Bibliotecas: são desenvolvidas em C/C++ e chamadas através de uma interface JAVA. As funcionalidades são disponibilizadas pelas Bibliotecas são acedidas através da *framework* aplicacional. Dentro das Bibliotecas estão presentes as que fazem a gestão das janelas, gráficos 2D e 3D, codecs de áudio e vídeo, base de dados, entre outras [24]

.

***Android Runtime*:** O ambiente do *Android* possui um conjunto de bibliotecas que fornecem todas as funcionalidades disponíveis nas bibliotecas JAVA. Estas bibliotecas aumentam as suas funcionalidades à medida que as versões do *Android* são lançadas. A máquina virtual *Dalvik* foi desenvolvida para executar máquinas virtuais de forma eficiente, executando ficheiros no formato Dalvik executable (*.dex*) e conseguindo otimizar a utilização da memória [26]

.

***Linux Kernel*:** O *kernel* 2.6 do *Linux* é utilizado pelo SO do *Android* e funciona como uma camada de abstração entre o *hardware* e o conjunto de software, sendo responsável pela gestão de processos, memória, gestão de rede e segurança do sistema [26].

O SO tem o seu constante desenvolvimento virado para o utilizador, tendo uma série de funcionalidades padrão incorporadas no próprio sistema:

- *Framework* de aplicações para reutilização de recursos;
- Suporte a diferentes tamanhos e densidades de ecrã, além de gráficos otimizados utilizando tecnologia 2D e 3D com base em especificação *OpenGL ES*;
- Base de dados SQLite;

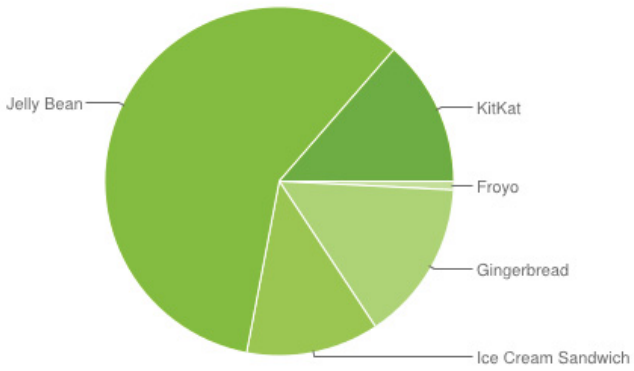
- Suporte a uma variedade de tecnologias GSM (*Global System for Mobile Communications*)/EDGE, CDMA *Code Division Multiple Access*), *Bluetooth*, 3G, 4G, WiFi (*Wireless Fidelity*) e NFC (*Near Field Communication*);
- SMS e (*MMS Multimedia Messaging Service*), incluindo GCM (*Google Cloud Messaging*);
- Suporte a *hardware* externos;
- Suporte a vários formatos de áudio, vídeo e imagem, inclusive Streaming;
- Suporte a *Tethering*, que permite que um dispositivo possa ser usado como ponto de acesso a *internet* por outro aparelho;
- Maquina virtual *Dalvik*, modificada para um funcionamento otimizado em dispositivos móveis;
- O SDK, com tudo o que é necessário para o desenvolvimento;
- *Google Play*, uma loja online de aplicações desenvolvido pela *Google*

[24]

Versões das Plataformas android

Tabela 3.1: Versões do SO *Android*

Versão	CodeName	API	%
2.2	*Froyo*	8	0,8%
2.3.7 2.3.7	Gingerbread	10	14,9%
4.0.3- 4.0.4	*Ice Cream Sandwich*	15	12,3%
4.1.x	*Jelly Bean*	16	29%
4.2.x		17	19,1%
4.3		18	10,3%
4.4	KitKat	19	13,6%



No que respeita a instalações das versões do *Android* nos dispositivos, pode-se analisar que a migração para versões mais recentes não ocorre tão rapidamente, como por exemplo nos computadores, este fator deve-se principalmente a questões relacionadas com o *hardware* dos dispositivos, que pode não estar adaptado. Neste momento as versão mais utilizada é a *Jelly Bean*, a mais recente versão a *kitKat* ainda tem valores baixos, apesar de ter avanços tecnológicos [27].

Tabela 3.2: API

Versão	**API**	***Version Code***
Android 1.0	1	*Base*
Android 1.1	2	*Base_1_1*
Android 1.5	3	*Cupcake*
Android 1.6	4	*Donut*
Android 2.0	5	*Eclair*
Android 2.0.1	6	*Eclair_0_1*
Android 2.1	7	*Eclair_MR1*
Android 2.2 - 2.2.3	8	*Froyo*
Android 2.3 – 2.3.2	9	*Gingerbread*
Android 2.3.3 – 2.3.7	10	*Gingerbread_MR1*
Android 3	11	*Honeycomb_MR1*
Android 3.1	12	*Honeycomb_MR2*
Android 3.2	13	*Honeycomb*
Android 4.0 – 4.0.2	14	*Ice_Cream_Sandwich*
Android 4.0.3 – 4.0.4	15	*Ice_Cream_Sandwich_MR1*
Android 4.1	16	*Jelly_Bean*
Android 4.2	17	*Jelly_Bean_MR2*
Android 4.3	18	*Jelly_Bean_MR1*
Android 4.4	19	*KitKat*

A plataforma *Android* fornece uma *framework* API para os programadores desenvolverem aplicações para a interação com o SO *Android*. A *framework* do *Android* tem por base:

- Um conjunto de pacotes e classes;
- Um conjunto de elementos e atributos XML (*Extensible Markup Language*) para declarar um ficheiro de configuração;
- Um conjunto de elementos e atributos XML para declarar e aceder a recursos;
- Um conjunto de Intenções;
- Um conjunto de permissões que as aplicações podem solicitar, bem como um reforço de permissões para o sistema.

Cada nova versão vem incluir atualizações e novas funcionalidades na plataforma *Android*, a estrutura da API esta projetada para que cada nova versão seja compatível com as anteriores, desta forma as partes mais antigas são substituídas, mas não são removidas (salvo pequenas exceções) para que as aplicações existentes as

possam utilizar [28].

Sistema de Ficheiros

O *hardware* utilizado nos dispositivos móveis para armazenamento são as memórias *flash*, que até a versão 2.3 *Gingerbread* utiliza o sistema de ficheiros YAFFS2 (*Yet Another Flash File System 2*), a sua utilização deve-se ao facto de ser um sistema aberto e preparado para memorias *flash* com um desempenho aceitável, foi o primeiro sistema preparado para memórias *flash* do tipo NAND (Not And), que funcionam a alta velocidade. Depois da versão 2.3 ocorreu uma migração no sistema de ficheiros, passou-se a utilizar o EXT4 (*Fourth Extend file System*) o principal motivo desta alteração é a falta de suporte do YAFFS2 para a multitarefa, com a introdução de processadores mais rápidos e com vários núcleos, isso passou a ser uma necessidade, melhorando também o desempenho no processamento de ficheiro com grandes dimensões e pela utilização de memórias *flash* do tipo eMMC (Embedded MultiMediaCard). Por vezes é possível encontrar outro tipo de sistema de ficheiros, devido ao fato de cada fabricante poder alterar o *Kernel* para adapta-lo ao tipo de *hardware* que vai utilizar [29][30].

3.2 Computação Forense em Dispositivos Móveis

3.2.1 Importância da forense nos *smartphone*

O smartphone foi dispositivo de comunicação móvel que mais evoluiu e desenvolveu-se nos últimos anos, aumentando de forma exponencial as capacidades e recursos de que dispõe. Dessa forma criou-se um novo mercado tanto ao nível das comunicações como das aplicações para os smartphones, essa evolução veio permitir que os *smartphone* conseguissem criar e aceder a diversos tipos de dados, dados esses que podem ficar armazenados nos dispositivos. Mas a sua crescente utilização de forma ambígua veio criar uma necessidade, a de desenvolver métodos para realizar uma análise forense aos diversos dispositivos. Para isso os investigadores têm de ser capazes de extrair e analisar os dados que são armazenados nos smartphones, assim torno-se necessário criar métodos e ferramentas que permitam a execução dessas tarefas de forma correta. Além disso, o rápido crescimento das tecnologias torna necessário que esses métodos sejam capazes de se adaptar a essa evolução.

3.2.2 Caso especial

Enquanto que os modelos de análise forense aplicados a *hardware* de computadores, como por exemplo, discos rígidos, tem procedimento já definido e estáveis, nos dispositivos móveis ainda existe muito trabalho a fazer sobre a técnicas de análise ao seu *hardware*. Os dispositivos móveis ainda são considerados fora do padrão tradicional de análise devido a sua heterogeneidade. Dentro de todas as investigações é necessário seguir os princípios básicos forenses. Mas existem dois princípios básicos que é importante seguir:

1. Ter o maior cuidado possivel com as evidência é manter o seu estado original;
2. O decurso de uma investigação deve ser compreensível e aberto a escrutínio. Na melhor das hipóteses, os resultados da investigação deve ser reprodutível por investigadores independentes.

Especialmente o primeiro princípio é um desafio no contexto dos dispositivos móveis, já que a maioria deles utilizam SO específicos e métodos de proteção de *hardware* que restrigem o acesso aos dados do sistema. A preservação de dados a partir de discos rígidos é na maioria dos casos, um procedimento simples e bem documentado. Um investigador remove o disco rígido do computador ou *notebook*, faz a conexão com o seu computador e com a ajuda de um *write blocker* e começa a analisá-lo com um *software* forense. Ao comparar estes procedimento com o dos dispositivos móveis torna-se claro que os procedimentos são diferentes. Quase todos os dispositivos móveis tem implementado um sistema para armazenamento próprio, com base no SO que utilizam, isso implica que cada investigador tem de definir formas de conseguir extrair os dados.

3.2.3 Operador da Rede Móvel

Para garantir que exista uma harmonização de nos regulamentos nos Estado Membros da União Europeia, a UE (União Europeia) emitiu em 2006 uma diretiva que visava criar regras para a conservação de dados gerados por serviços de comunicação eletrotécnicas. A diretiva permitia aplicar a lei para aceder aos dados de tráfego relativos aos utilizadores. De acordo com esta diretiva eram armazenados por um período entre 6 meses a 2 anos os seguintes dados, relativos a dispositivos móveis:[31]

- os números de telefone de origem e de destino;
- IMSI (*International mobile subscriber identity*) de quem telefona;

- IMEI (*International Mobile Equipment Identity*) de quem telefona;
- a IMSI do destinatário do telefonema;
- a IMEI do destinatário do telefonema;
- a data, hora e a duração de uma comunicação;
- o tipo de comunicação (mensagem SMS, MMS ou telefonema);
- a localização de equipamentos móveis de comunicação (por exemplo, dados de GPS ou pelo menos a localização da célula móvel que foi usado).

De acordo com a directiva da UE [31], e obrigatório estarem disponíveis para as autoridades nacionais "competentes"em casos específicos", para fins de investigação, detecção e repressão de crimes graves, como definido por cada Estado-Membro no seu direito nacional".

Mas surge um Acórdão do Tribunal de Justiça de 8 de Abril de 2014[32] que veio inviabilizar a diretiva de 2006/24/CE, levantado uma série de duvidas sobre o âmbito da sua aplicação as autoridades portuguesas. Estando os operadores de serviços de comunicação a conservar dados de tráfego e de localização para efeitos da sua eventual transmissão as autoridades, sendo este processo contrário aos princípios enformadores do Direito da UE [33][34].

3.2.4 *Mobile Forensic*

Uma das definições que podemos encontrar sobre *"Mobile phone forensics"* encontra-se em, *"Mobile phone forensics is the science of recovering digital evidence from a mobile phone under forensically sound conditions using accepted methods."* NIST [35].

As investigações a dispositivos móveis já são comuns na área da Computação Forense, consequência das novas forma como são utilizados. Mas essa investigação implica métodos adaptados aos dispositivos móveis. Embora muitos dos métodos tenham sido desenvolvidos inicialmente para serem aplicados em computadores, esses métodos foram sendo adaptados para as características dos dispositivos. Segundo o Departamento de Justiça dos Estados Unidos toda a investigação é distinta e tem conjunto próprio de circunstâncias, uma abordagem de processo definitiva é difícil de ser prescrita.

No entanto, a maioria das metodologias tem os mesmos pontos chaves, embora com aspetos diferentes [22]. É importante referir que perícia em computadores e um assunto com particularidades diferentes, a necessidade de interação nos dispositivos móveis é um ponto importante a destacar, principalmente se levarmos em conta que para cada SO é necessário que haja um entendimento dos procedimentos específicos, e isso é um ponto crucial para facilitar a perícia. Sem essa abordagem, a investigação para um dispositivo móvel pode não ter resultados significativos.

3.3 Metodologia de Análise em Dispositivos *Android*

3.3.1 Preservação/Apreensão do Dispositivo Móvel

A Preservação/Apreensão é o ponto de partida de todo o processo e para que se consiga obter os melhores resultados é preciso que se tenha o máximo cuidado possível para que seja evitada a possibilidade de perda ou alteração da prova. A preservação envolve pesquisa, reconhecimento, documentação e recolha de evidências, com o intuito de utilizar as provas com sucesso, seja em um tribunal ou em um processo informal, por isso as evidências devem ser preservadas. Um erro ao preservar as provas em seu estado original poderia comprometer a investigação e possivelmente ocasionar a perda de informações importantes sobre o caso [35]. A ACOP sugere os seguintes procedimentos:

- Antes de apreensão, considerar outros tipos de provas, como o DNA ou impressões digitais, que podem ser adquiridos no dispositivo e sem o devido cuidado podem ser destruídas ou contaminadas;
- Desligar o dispositivo é aconselhável, por causa do potencial de perda de dados se acabar a bateria ou houver atividades na rede a qual o dispositivo esta conectado, fazendo com que os registos de chamadas ou outros dados sejam substituídos (Neste caso, a avaliação quanto a restrição de acesso ao aparelho deve ser feita, por exemplo, se uma autenticação de acesso vai ser necessario após o encerramento do dispositivo);
- Se o dispositivo permanecer ligado por algum motivo, deve ser mantido carregado e sem interações, então, desligado antes do transporte adequado;
- Para impedir o funcionamento acidental no transporte, o telefone deve ser acondicionado de modo a não sofrer nenhum acidente;

- Ao transportar os dispositivos, eles devem ser adequadamente protegidos contra quedas, choques e temperaturas extremas;
- O dispositivo deve ser colocado em um saco de provas, vedado, para restringir o acesso;
- Deve-se também observar os acessórios do equipamento, como carregadores, embalagens e manuais;
- Deve-se entrevistar os donos dos equipamentos, a fim de obter possíveis senhas de acesso e outras informações relevantes [14].

Os equipamentos associados ao dispositivo móvel, como memórias removíveis, cartões SIM (*Subscriber Identity Module*), computadores pessoais que podem ter sido usados para sincronização, podem ser mais valiosos do que o próprio dispositivo. É importante observar também que neste processo de apreensão e preservação todas as provas e o local devem ser documentadas e/ou fotografadas.

Um registo de todos os equipamentos visíveis deve ser criado, devendo ser fotografados evitando modificar o ambiente onde foi encontrado. O ecrã também pode ser fotografada e, se necessário, documentado. É importante, evitar entrar no local de buscas com dispositivos *Wi-Fi e Bluetooth* ativados, para não ocorrer interações indesejadas com o dispositivo móvel encontrado. Para evitar também o contacto do dispositivo com redes móveis, deve-se colocar o dispositivo em modo de vôo, avião ou *offline*. Se o dispositivo estiver ligado, deve-se configurá-lo para um modo de funcionamento sem conexão, evitando a transmissão de dados, receção de chamadas ou mensagens após a apreensão do equipamento. Se não for possível realizar exames em local com isolamento, sem sinal de cobertura da rede e o telefone tocar ou receber algum dado seja SMS ou interação com dados da *internet*, é necessário que o registo dessa ocorrência seja feito, com a maior recolha de detalhes possíveis é recomendado jamais atender ou interagir com o dispositivo [35] [22].

3.3.2 Aquisição/Extração de Dados em dispositivos móveis

Esta etapa consiste no processo de criação de imagens forenses ou outros processos para a extração de informações do dispositivo móvel, seus periféricos ou equipamentos e memorias removíveis. Sendo este o momento mais técnico, onde os conhecimentos por parte do perito vão ser mais importantes. Sendo um processo complexo e que necessita de um especialista com conhecimentos específicos sobre a

plataforma *Android*, pois poderá ser necessário uma intervenção manual para viabilizar a aquisição de dados. Para demonstrar os processos e decisões envolvidos na extração dos dados por parte do investigador ao realizar uma análise a dispositivo com SO *Android* temos no diagrama 3.2.

Aquisição e preservação dos dados do cartão e do smartphone

A zona 1 identifica os passos da etapa inicial de extração de dados. Estando o dispositivo desligado, a preocupação vai ser extrair os dados do cartão de memória, que possa ser removido do dispositivo (e.4), substituindo o cartão original por outro cartão de memória que contenha uma cópia dos dados originais. Alguns modelos podem não ter cartão de memória tendo em alternativa memórias internas (e.3). Para a cópia dos dados pode-se utilizar ferramentas forenses e gerar o *hash* dos dados duplicados. O próximo passo consiste em verificar se é possível isolar o dispositivo da rede do operador (e.5), por meio de uma sala com isolamento de sinais eletromagnéticos, a fim de evitar a comunicação do dispositivo com fontes externas antes de ligá-lo (e.7). Caso não seja possível isolar fisicamente o dispositivo da rede, o investigador deve ligá-lo (e.6) e imediatamente colocá-lo em modo de avião (e.8).

Aquisição dos dados de *smartphone* sem controle de acesso

Na zona 2, são identificados os processos de extração de dados em dispositivos *Android* onde o controle de acesso esteja ativado, situação que levanta dificuldades para o investigador. Com o dispositivo ligado, é possível verificar se existe algum controle de acesso ativado. Não estando bloqueado (e.9) ou após desbloqueá-lo (e.11 e e.20) ou ainda se ele tiver acesso de depuração ADB (Android Debug Bridge) com permissões de super utilizador (e.19), caso não tenha sido realizada a extração dos dados do cartão) (e.10), este será o momento de extraí-los. Deve-se utilizar bloqueio de escrita, e, se possível, clonar o conteúdo para um cartão do investigador que irá substituí-lo no dispositivo (e.12). Deve ser avaliada a substituição do cartão original em situações em que não seja possível a sua remoção, ou quando se tem a necessidade de remover a bateria para retirá-lo não sendo desejável, desligar o dispositivo. Nestas situações é justificável a utilização do cartão original para realização da investigação após ter sido integralmente copiado.

Verificação das permissões de super utilizador

Em seguida, é preciso verificar se o dispositivo tem permissões de super utilizador (e.13). Estando com essas permissões ativadas, deve-se copiar o conteúdo integral

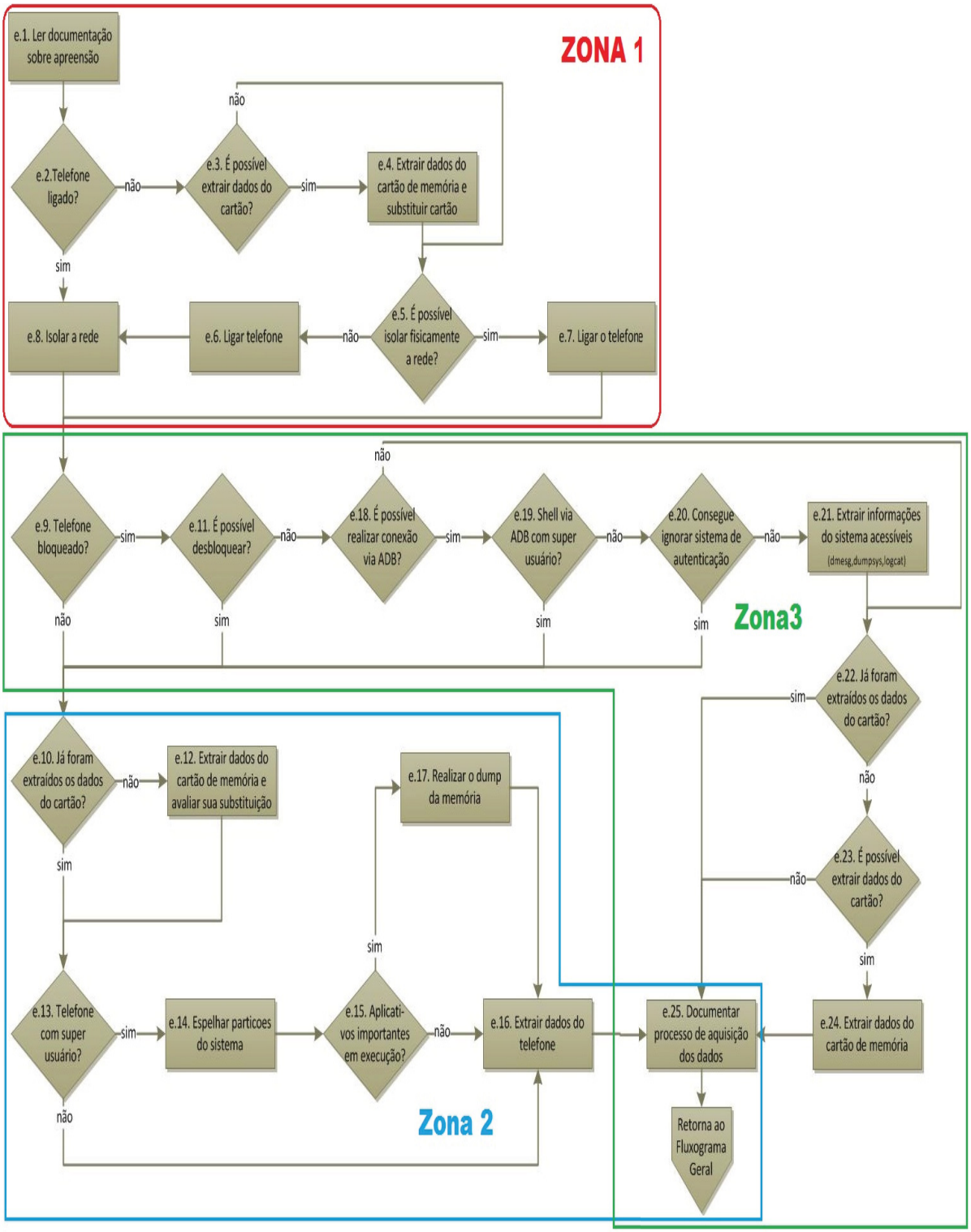


Figura 3.2: Modelo Forense

das partições do sistema (e.14). O processo de copiar as partições do sistema com permissões de super utilizador é a forma mais completa de obter os dados. É possível ao investigador copiar integralmente as partições do sistema por meio dos comandos cat ou dd. É importante esclarecer que, com as técnicas atuais, os ficheiros gerados com o conteúdo das partições do sistema (imagens) serão gravados no cartão de memória instalado no aparelho. Depois, quando for o caso (e.15), deve extrair os dados em memória dos processos que se encontram em execução (e.17), para ter acesso a dados como *password* e chaves criptográficas.

Extração dos dados do telefone

Com os dados do cartão de memória original preservados, assim como do sistema quando for possível, inicia-se o processo de extração de dados do dispositivo (e.16). Neste processo o cartão que se encontra no aparelho deve estar devidamente preparado para realizar a extração dos dados do dispositivo, com espaço suficiente para armazenar os dados que serão extraídos. Caso o sistema tenha permissões de super utilizador, é possível copiar os ficheiros da base de dados, cache das aplicações e os ficheiros de configuração do sistema, e até mesmo realizar uma inspecção visual, a fim de facilitar o acesso a estas informações na etapa de exame, assim como arquivos gerados nos processos anteriores a exemplo dos ficheiros de *Dump* de memória.
Isso porque para a análise posterior dos dados extraídos, o investigador deve ter um ambiente de exames com ferramentas para montar imagens com suporte ao sistema de ficheiros utilizado no dispositivo, geralmente o YAFFS2. A criação desta redundância poderá ser útil no momento dos exames, principalmente em situações que seja necessário aprofundar a análise das partições do sistema. Para as aplicações que estejam ativas no sistema, uma simples inspeção visual pode fornecer informação que seria de difícil acesso por meio da análise da imagem gerada. Outra opção é utilizar as ferramentas para extração de dados dos dispositivos, como o *Cellebrite UFED*, (*Open Source Android Forensic Agent*), *AFLogical*, entre outros [36] [37].

Para realizar a extração dos dados desta forma é necessária a instalação de um ou mais aplicativos no dispositivo. Esta instalação deve ser realizada através da ferramenta ADB, com o dispositivo configurando para aceitar conexão USB no modo de depuração. Outra forma de instalá-los é pelo gestor de aplicações que permita a navegação no cartão de memória, onde eles deverão estar armazenados, com o dispositivo configurado para aceitar instalações de aplicações que não são do *Android Market*. Ao instalar uma aplicação forense no dispositivo, os dados são modificados. Entretanto, esta abordagem é interessante para se conseguir extrair todos os dados

acessíveis ao utilizador no dispositivo, devendo assim ter acesso manual ao telefone. Para isso, a aplicação forense terá no ficheiro "*AndroidManifest.xml*" todas as permissões necessárias para realizar a extração.

A fim de complementar a extração dos dados no dispositivo (e.16), o investigador poderá navegar pelo sistema *Android* sob a perspetiva do seu proprietário, podendo inclusive observar a forma com que ele utilizava as aplicações instaladas, conferindo as últimas ligações efetuadas e recebidas, analisando as mensagens de e-mail e de texto, dentre outras atividades. Inclusive, recomenda-se ao investigador comparar os dados obtidos a partir da extração do aplicação forense com os dados efetivamente armazenados no dispositivo com a finalidade de auditar as informações recolhidas e complementá-las quando necessário outra questão prende-se com questões de desenvolvimento das aplicações por causa das versões do SO *Android*.

A aquisição dos dados de smartphone com controle de acesso

No caso do dispositivo ter com restrição de acesso (e.9), em que não foi possível desbloqueá-los (e.11), o investigador pode ainda tentar realizar uma conexão USB com o computador e tentar aceder via ADB (e.17). Se o acesso via ADB não for possível, não será viável ao investigador extrair os dados contidos no sistema do dispositivo, a não ser que utilize técnicas de extração mais agressivas como acesso por *bootloader*, restando ao investigador realizar a extração do cartão de memória, caso ainda não tenha sido realizada (e.22, e.23 e e.24).

Caso contrário, o investigador poderá tentar obter uma *shell* com permissões de super utilizador(e.19), e se possível, prosseguir com a extração dos dados. Caso seja possível utilizar a ferramenta ADB, mas não se consiga permissões de super utilizador, poderá ainda tentar configurar o sistema para ignorar o sistema de autenticação (e.20), instalando uma aplicação para este fim. Para isso é necessário que a senha da conta *Google* esteja gravada no dispositivo *Android* e ter acesso à *Internet*, o que é desaconselhável.

Desta forma, recomenda-se obter o aplicação a partir de outro dispositivo *Android* e instalá-lo via ADB no dispositivo examinado. Se não for viável ignorar o sistema de autenticação, ainda é possível obter informações das ferramentas de *log* do sistema (e.21), a exemplo do *dmesg*, *dumpsys* e *logcat*, dependendo do que se esta a investigar, poderá fornecer algum auxílio no exame, para depois tentar realizar a

extração do cartão de memória, caso ainda não tenha sido realizada (e.22, e.23 e e.24)

Considerações sobre a aquisição

Todo o processo de extração dos dados deve ser documentado (e.25) tendo em conta os passos dados, descrevendo também as informações sobre o sistema *Android*, como sua versão e *kernel*. A correta documentação de todos os procedimentos será essencial para o processo de exame, sendo necessário descrever no relatório todos os procedimentos realizados com a finalidade de esclarecer a forma como os dados foram adquiridos.

3.3.3 Exame e Análise

Os resultados obtidos nesta fase são o resultado da aplicação de metodologias estabelecidas com bases científicas. Deve-se então descrever as informações encontradas, incluindo a origem, o estado e significado delas. O investigador deve ter bem definido quais são os objetivos da investigação, para isso é importante ter um conhecimento global de toda a investigação para conseguir dar uma resposta às questões que se pretende encontrar. E importante fazer uma caracterização do caso e a uma análise prévia para determinar o que será procurando, por exemplo para determinar se foram feitas chamadas depois de um determinado acontecimento, focar a procura para depois da data e hora do acontecimento.

Na análise dos dados extraídos, o investigador deve estara atento para configurações como data e hora, linguagem, configurações regionais, contactos, agenda, mensagens, chamadas (realizadas, recebidas e não atendidas), imagens, vídeos, áudios, dados sobre localização e mensagens multimédia. Registos como históricos da navegação *web*, documentos, informações do GPS e dados criados por aplicações instaladas no dispositivo. Os dispositivos com o SO *Android*, para além da capacidade de armazenamento têm a possibilidade de instalar variais aplicações que tem funcionalidades associadas à *Cloud Computing*. Esta característica, pode ser interessante numa investigação mais aprofundada, ao procurar informações que estão além da barreira física imposta pelo dispositivo. O exame dos dados extraídos pode variar da forma como os dados foram obtidos, se foram extraídos a partir de uma ferramenta forense, deve-se analisar a forma como os resultados são apresentados, observando o relatório gerado e os arquivos gerados e recuperados. Geralmente, as ferramentas específicas para uma determinada plataforma produzem bons resultados, uma vez

que o processo é automatizado. No entanto, na fase de aquisição, o investigador deve realizado uma comparação daquilo que foi extraído pela aplicação com as informações que constam no dispositivo. Se os dados extraídos foram obtidos a partir de uma imagem do sistema por meio de um acesso de super utilizador, o investigador pode utilizador editores hexadecimais e ferramentas forenses usadas para análise do cartão de memória, e outras técnicas periciais [35] [22].

3.3.4 Formalização e Documentação

Na etapa de formalização será criado um resumo detalhado, de todos os procedimentos importantes usados nas fases anteriores, junto com as conclusões obtidas na investigação. Será feito um registo cuidadoso das ações, procedimentos adotados e observações técnicas durante as fases de preparação, aquisição e exame são importantes e vão fazer parte deste relatório final.

Ao elaborar um relatório de um exame de um dispositivos móvel, devem ser descritas as características do dispositivo examinado, como: número IMEI, marca e modelo do equipamento, operador e número de cartão SIM e, caso exista, a marca, modelo, capacidade e tipo de cartão de memória. O relatório final deve incluir também todas as informações necessárias para identificar o caso, os resultados dos exames e evidências encontradas, identificar o perito, identificação do requisitante dos exames, a data de recebimento da requisição e a data de criação do relatório [22].

3.3.5 *Software* de Extração de dados

É possível instalar aplicações nos dispositivos *Android* que têm a finalidade de recolher dados do utilizador, o processo de recolha pode variar conforme a aplicação é desenvolvida, mas todos eles necessitam de ser instalados no dispositivo através do ADB ou por instalação do gestor de aplicações. O funcionamento consiste no acesso aos dados por "*content providers*" e pelas correspondentes permissões de acesso aos dados, devido a este procedimento, e possível aceder aos dados armazenados e protegidos a partir de aplicações instaladas. Algumas das vantagens que resultam da utilização de *software* de extração de dados é que necessitam de pouco conhecimento técnico, não são invasivos para com o dispositivo, não é necessario *hardware* especial para ler os dados a partir do dispositivo. Também é possível recuperar os dados apagados, desde que ainda estejam visíveis na base de dados. No entanto, a grande desvantagem é que é possível que alguns dados sejam modificados por via da copia

ou instalação dos *softwares*, algo que pode comprometedor no processo de exame, mas que devidamente justificado por ser útil, dependendo dos dados que se pretende obter. Como exemplo temos cinco software de extração, *Cellebrite UFED* , *Secure View 3*, *Paraben*, ADEL e AFLogical-OSE [36] [37].

Capítulo 4

Estado da Arte e Hipótese de Investigação

4.1 Enquadramento do Estado da Arte

A análise do estado da arte foca-se em quatro aspetos, que vão de encontro ao trabalho desenvolvido para se realizar uma investigação forense a um dispositivo móvel com SO *Android*. Sendo que os procedimento forenses para SO *Android* são recentes e por vezes pouco definidos, se compararmos com procedimentos já bem definidos para os computadores pessoais. O primeiro aspeto que se analisou foi a capacidade de se fazer uma investigação em qualquer uma das versões do SO *Android*. O segundo passa pela definição de metodologias eficientes para o processo de análise forense. No terceiro analisa-se o *software* que se utiliza na análise forense a dispositivos móveis. Por ultimo, o quarto aspeto a forma como se pode analisar os dados recolhidos durante a extração dos dados, sendo que neste caso só referimos o artigo [38], porque não tem sido um aspeto muito referido nos trabalhos desenvolvidos.

O site *Android Developers* [39] é o principal repositório de informação sobre o SO *Android* incluindo ferramentas de depuração e de desenvolvimento. O site contém informações sobre *scripts*, como desenvolver aplicações *Android* ("*apps*") e como realizar depuração de aplicações existentes. As ferramentas de desenvolvimento do *Android* estão disponíveis como um plug-in para o Eclipse ambiente de desenvolvimento integrado (IDE), uma plataforma de desenvolvimento JAVA. De referir que para os investigadores forenses a ferramenta SQLite3 permite a análise das bases de dados SQLite utilizados pelo SO *Android*. Existem outros recursos como o *Traceview*, que produz visualizações gráficas de análise de dados de *log*, que podem ser

gerados a partir de uma aplicação *Android*. O site tem diversa documentação que explica os conceitos do SO *Android*, de referir a importância da documentação sobre as varias versões o SO. Antes de qualquer investigação é fundamental que o investigar tenho um conhecimento profundo da estrutura e do funcionamento do SO, sendo este site o ponto de partida para se adquirir esse conhecimento.

Hoog é uma das grande referências que existe sobre Analise Forense, no livro de *Alexander Hoog*, que faz referência ao fato de que o SO *Android* é executado em muito mais do que em *smartphone* , tais como *tablet*, PCs, eletrodomésticos e outros dispositivos móveis. Isso demonstra a necessidade de existir conhecimento básico do funcionamento do SO *Android* por parte dos investigadores.

Hoog explica como o sistema operacional *Android* é baseado no *Linux* 2.6 *kernel* e gerido pela OHA, tornando-se uma plataforma de código aberto que é atraente para desenvolvimento, propõe-se quatro abordagens para a investigação forense no SO *Android*:

- Análise de cartão SD
- Aquisição lógico
- Aquisição físico
- Chip -off

O cartão SD é formatado com um sistema de ficheiros FAT32 padrão, e pode ser analisado de forma normal, em *write-blocking* as imagens do cartão vão ser analisada por uma ferramenta forense ou editor hexadecimal. *Hoog* recomenda a aquisição lógica de um dispositivo *Android* como a melhor opção para os investigadores. Tendo como referência uma ferramenta de código aberto desenvolvida pela *viaForensics* de onde se pode obter as seguintes informações, como se pode ver na tabela 4.1.

Também desenvolveu um método para fazer a extração física dos dados no dispositivo *Android* no formato YAFFS2. A desvantagem é que existem poucas ferramenta que suportam este formato de ficheiros. Isto significa que a análise física terá de ser feita manualmente [40].

Tabela 4.1: Informação extraída pela aplicação viaForensics

Histórico do *Browser*	Registro de chamadas
Contactos	*External Image Media*
External Image Thumbnail Media	MMS
External Media, Audio,	SMS
External Videos	Pessoas
Organizações	Contactos Grupos
Contactos telefónicos	Contactos Extensões
MMSParts	Definições dos contactos
Lista de todas as aplicações instalados e versão	

4.2 Independência do tipo de plataforma *Android*

Em *Votipka, Vidas, Christin et. al.* utiliza um método de extração físico para ultrapassar a questão da independência da versão do SO *Android*. Esta solução define quatro características fundamentais para a recuperação da imagem do dispositivo *Android*: cabeçalho de imagem, *Kernel*, *ramdisk e flashing tool*. Esta metodologias permite também recolher a informação contida na memória do dispositivo. Um ponto importante e fundamental é que durante todo o processo se consiga extrair o maior conjunto de dados causando o mínimo da danos aos dados extraídos, sempre com o objetivo de preservar as provas recolhidas. Baseando-se no método de [41] foi possível preservar os dados do utilizador, recolha total, com exatidão, determinismo, preservação da prova e usabilidade e reprodutibilidade do processo através da técnica de criação de imagem. De referir que para conseguir extrair a imagem é necessário ter acesso ao dispositivo via USB pelo ADB [42].

Grover, propõe uma solução para contornar as restrições que existem nas várias versões, que consiste num sistema de monitorização para *smartphones Android* que foi desenvolvido para fazer uma recolha continua de dados. O protótipo do sistema não requer privilégios de *root*, nem necessita de fazer uma exploração à arquitetura *Android*, aumentando assim a interoperabilidade entre os dispositivos *Android* e evitando que o sistema seja classificado como *spyware*. As contribuições desta pesquisa incluem o lançamento do primeiro tipo de solução de monitorização *Android*. A aplicação *DroidWatch* é um protótipo desenvolvido para ser um sistema automatizado composto por uma aplicação *Android* e um servidor. Depois de ser instalado no sistema *Android*, não sendo necessário permissões de *root, DroidWatch* consegue fazer uma recolha continua de informação, armazenar esses dados e fazer

a transferência dos mesmos para um servidor *Web* remoto. Sendo capaz de recolher um conjunto de dados que não foi possível recolher com outros *software* [43].

Vidas, Zhang e Christin apresentam no seu artigo um processo de recolha de dados generalizado para que funcione em diversos dispositivos com SO *Android* [41].

4.3 Metodologias de Análise

Este trabalho desenvolvido por *Son, Lee, Kim, James, Lee, Lee*, utiliza como método de extração de dados no SO o '*Recovery Mode*' designado por CRMI (*custom recovery mode image*). São avaliadas as variáveis do modo de recuperação do SO, que potencialmente pode comprometer a integridade dos dados no momento da aquisição de dados. Com base na análise realizada, foi desenvolvida uma ferramenta de aquisição de dado, que garante a integridade dos dados adquiridos. Sendo posteriormente demonstrado através de um estudo a capacidade que a ferramenta tem de manter a integridade dos dados. Os testes foram efetuados para o sistema de ficheiros YAFFS2 e Ext4, o único problema neste método tem a ver com a dificuldade de obter o firmware original do dispositivo, o que leva a utilizar o CRMI para a partição de recuperação em vez da partição de *Boot* [44].

Stüttgen e Cohen focam-se na aquisição da memória volátil do dispositivo. Sendo referido o problema da utilização de agentes forenses em sistema comprometidos, pois o sistema pode fornecer informações falsas, devido às suas vulnerabilidades como a instalação de *malware*. Embora seja referido que a técnica não é perfeita, as contramedidas apresentadas não mostram o nível de complexidade exigido para subverter com os processo de aquisição. Por exemplo, para combater com sucesso a técnica de aquisição, o *rootkit* pode precisar de manipular as tabelas de páginas do *kernel* directamente, aumentando a complexidade e reduzindo a capacidade do *rootkit* para manter o sistema estável após ter sido comprometida. Embora não se tenha encontrado problemas com todas as ferramentas testadas, foi possível adicionar a técnica a ferramenta de aquisição *WinPmem Open Source* [45].

Grover, desenvolve no trabalho uma metodologia onde se instala uma aplicação DroidWatch, que tem a função de ser um agente forense que recolhe os dados num processo continuo e os envia via HTTP (*Hypertext Transfer Protocol*) a cada 2 horas para um servidor que processa os dados recebidos. Esta solução tem a vantagem de conseguir recolher dados em tempo real. Como solução forense tem um funcionamento diferente do normal, a captura dos dados é feita com o sistema em execução pelo utilizador do dispositivo, tem a vantagem de conseguir obter mais dados, princi-

palmente de *log* do sistema e praticamente em tempo real. Mas a grande dificuldade será a técnica utilizada para instalar a aplicação [43].

Grispos, Glisson e Storer surgem com um artigo que foca-se numa questão recente, o armazenamento de Informação na *Cloud*, algo que não se enquadra na tradicional análise forense, devido aos dados não estarem guardados no dispositivo, mas que surge como uma forma de armazenar e de aceder a dados nos dispositivos móveis. Sendo apresentado como um novo desafio para a comunidade Forense. Foca-se assim, a análise nos registos que possam ficar da utilização desses serviços de Armazenamento, no lado do utilizador. Primeiro, fornece uma avaliação inicial sobre a medida em que os dados de armazenamento na cloud são armazenados nos dispositivos do lado do cliente. Este ponto de vista funciona como um *proxy* para os dados armazenados na cloud. Em segundo lugar, ele fornece documentação sobre os artefatos que poderiam ser úteis numa investigação forense digital para um serviço da cloud. A recuperação de dados a partir dos dispositivos móveis, pode, em alguns cenários proporcionar o acesso a outros dados armazenados na *cloud*. Do ponto de vista do cliente, ele pode vir a dar uma visão parcial dos dados, sem acesso ao fornecedor de dados. A recuperação dos dados dependente de dois fatores. Primeiro, o aplicação de armazenamento da *cloud* ter sido usado para visualizar os ficheiros na nuvem. Em segundo lugar, o utilizador não tentou apagar a cache dos ficheiros exibidos recentemente. O modelo de análise utilizado demostrou ser possível recuperar dados que estão armazenados em serviços da *Cloud* [46].

Racioppo e Murthy escrevem um artigo, dirigido para uma análise forense a um dispositivo especifico, "*HTC Incredible*" com SO *Android*, criando uma metodologia de análise especifica. O processo consiste na extração física do conteúdo do dispositivo e posterior análise. Sendo possivel fazer uma análise lógica aos dados, no artigo foi dado muita importância à base de dados em SQLite, atravês do *SQLiteManager* [47].

Goel, Tyagi e Agarwalr definem neste artigo um modelo de processo de análise forense para os Smartphones. Tendo com base as metodologias forenses digitais, adaptam esses métodos, para serem aplicados nos *smartphone*. De realçar um ponto importante, a sua capacidade de se adaptar aos avanços tecnológicos que estes equipamentos sofrem. O modelo proposto explora os diferentes processos envolvidos na investigação com base num modelo de 14 fases, de modo a orientar o investigador para que se consiga obter o maior numero de dados [48].

Simão, Sícoli, Melo, Deus e Júnio propõem uma metodologia que tem por base a tradicional análise forense ao PC, mas que foi adaptada para ser aplicada aos dispositivos móveis, tendo em atenção as especificações da plataforma *Android*. É

proposto um fluxo de trabalho que abrange os diferentes cenários que um analista pode encontrar durante o processo de extração dos dados, sendo possível recuperar o maior número de informação e da forma mais segura [49].

Vidas, Zhang e Christin escreveram um artigo onde a principal contribuição consiste na proposta de uma metodologia em que se utilizam métodos especiais de inicialização dos equipamentos, permitindo utilizar um método definido por *recovery bootimg*, que garante uma elevada probabilidade de que os dados recolhidos não estão corrompidos. Depois de explorar os modos especiais de fazer *boot* aos dispositivos *Android* e de analisar o sistema de partições foi possivel definir o modo que criação de uma imagem do sistema e assim, proceder à sua inicialização. Tendo sido possível repetir a técnica em vários equipamentos. Mas não foram feitos testes de performance à solução proposta [50].

Este artigo implementa um sistema de análise que faz a aquisição dos dados através da *cloud* para realizar a análise forense e gerar relatórios da análise. O procedimento é o do NIST, os dados são adquiridos e enviados para *web* para serem tratados pelo sistema. Uma das vantagens será a facilidade com que se obtém os dados da memória volátil. A análise da informação recolhida é um ponto pouco valorizado, são poucos os artigos de fazem referência a esse processo, mas o este trabalho pela sua lógica de funcionamento permite a elaboração de relatórios dos dados, o que facilita o trabalho do investigador. Os dados recebidos via web são apresentados no browser e facilmente exportados para elaborar os relatórios [38].

Neste artigo que se foca na questão de contornar as medidas de segurança que os equipamentos podem ter ativados, neste caso o bloqueio de ecrã, que são cada vez mais utilizados como forma de proteger o acesso ao equipamento. Através de uma análise de resíduos que ficam no ecrã, por exemplo para orientação do ecrã, pela iluminação do ecrã, demonstra-se que é possível contornar estas medidas de segurança e fazer um ataque ao dispositivo.[51]

4.4 Software de Análise Forense

Son, Lee, Kim, James, Lee, Lee fazem referência neste artigo, a uma Ferramenta, *Android* Extractor, desenvolvida em C++ que pretende garantir a integridade dos dados. O processo pretende ser automatizado para todos os sistemas de ficheiros do SO *Android*, mas só foi testado em YAFFS2 e Ext4. Esta ferramenta vai possibilitar fazer uma extração física criando uma imagem dos dados do dispositivos e seguindo um procedimento de sete passos até se concluir o processo de investigação [44].

Para realizar a análise forense *Racioppo e Murthy* fazem referência a várias ferramentas, cada uma utilizada para realizar uma função especifica. Para realizar a aquisição dos dados do cartão SD utilizou-se o *software AccessData FTK Imager*, que criou uma imagem dos dados do cartão original que posteriormente é gravada noutro cartão para ser analisada num leitor de cartões SD. Para se obter as permissões de *root* utiliza-se o *UnrEVOked* um *sofwtare* que se instala no pc e se liga ao dispositivo de forma a se obter as permissões de *root*. Para criar uma imagem do sistema utiliza-se a ferramenta dd após se ligar ao dispositivo via ADB por linha de comando. Para fazer a análise da imagem utiliza-se o *software* scalpel. Para aceder a base de dados utilizou-se o *SQLiteManager* onde se pode obter toda a informação disponível na base de dados [47].

Yates apresenta um conjunto de ferramentas forenses que podem facilitar a análise ao investigador, tendo em conta a diversidade de fabricantes e modelos existentes. Foi feito um resumo para os analistas compararem as funcionalidades das ferramentas e as adequarem ao tipo de análise que se pretenda realizar. Inclui também algumas indicações das funcionalidades que devem ser incluídas nas aplicações [52].

Azadegan, Yu, Liu, Sistani e Acharya apresentaram um artigo onde consta uma lista das ferramentas e características para a aquisição através dos seguintes métodos:

- Aquisição manual - *Paraben Device Seizure*
- Extração física – XRY *Complete*
- Extração lógica - *SecureView*

Através dos testes realizados com as técnicas anti-forenses foi possível demonstrar as limitações das ferramentas durante o processo de obtenção dos dados. Este *software* segue os procedimentos definidos pelo NIST para análise forenses a dispositivo móveis [53].

Lee, Yang, Chen, e Wu escreveram um artigo em que são referidas em várias ferramentas forenses que podem ser usados para obter informações de um dispositivo móvel *Android* como *Paraben*, Cellebrite, *Susteen SecureView* e *viaForensics*. A ferramenta e a metodo lógico da *viaForensics* serão avaliados nesta pesquisa. A *viaForensics* desenvolveu uma solução que consiste na instalação de uma aplicação que permite a extração dos dados para o cartão SD [54].

4.5 Hipótese de Investigação

A investigação forense a um dispositivo móveis com SO *Android* pode ter varias condicionantes, desde a versão do SO até sistemas que bloqueiam o acesso ao dispositivo, entre outros. Para fazer uma análise forense é necessário fazer uma extração dos dados, estão definidas duas formas de fazer essa extração, Física e Lógica.
A maioria dos artigos analisados no estado da arte apresenta soluções de extração físicas onde a analise e feita a totalidade dos dados extraídos. Esta técnica funciona bem e por vezes é a forma de conseguir obter informação que foi apagada, mas tem a desvantagem de não se focar num conjunto especifico de informação. Por outro lado a, Extração Lógica feita por um agente forense, apresenta-se como uma mais valia a forma mais direta de obter informação, vai fazer uma pesquisa com base no que o utilizador definiu no agente, SMS, Registo de Chamadas, Contactos, entre outras informações que o agente forense pode extrair. O ponto comum que inviabilizaria as duas formas de extração é quando não é possível fazer uma conexão ao dispositivo via ADB.
A questão que se coloca, é a possibilidade de desenvolver um sistema de analise forense em dispositivos móveis que seja independente da versão do SO *Android* e que consiga gerar automaticamente relatórios para o exterior do dispositivo em formato PDF (*Portable document format*) com a informação que o investigador procura?

4.5.1 Adaptação da Metodoloia de Análise Forense para SO *Android*

Na dissertação de André Simão é apresentada uma metodologia que descreve todos os procedimento que se pode ter de utilizar durante uma aquisição dos dados. Para além de ser descritos todos os passos desde a apreensão até a elaboração do relatórios. Sendo uma metodologia bem estruturada e funcional optou-se por utiliza-la durante o processo de análise forense ao dispositivo móvel pelo agente forense. Na imagem 4.1 são demonstrados os procedimentos, que se seguem até se conseguir obter os dados, de referir que aplicamos esta metodologia sempre que é possível obter conexão com o dispositivo [55].

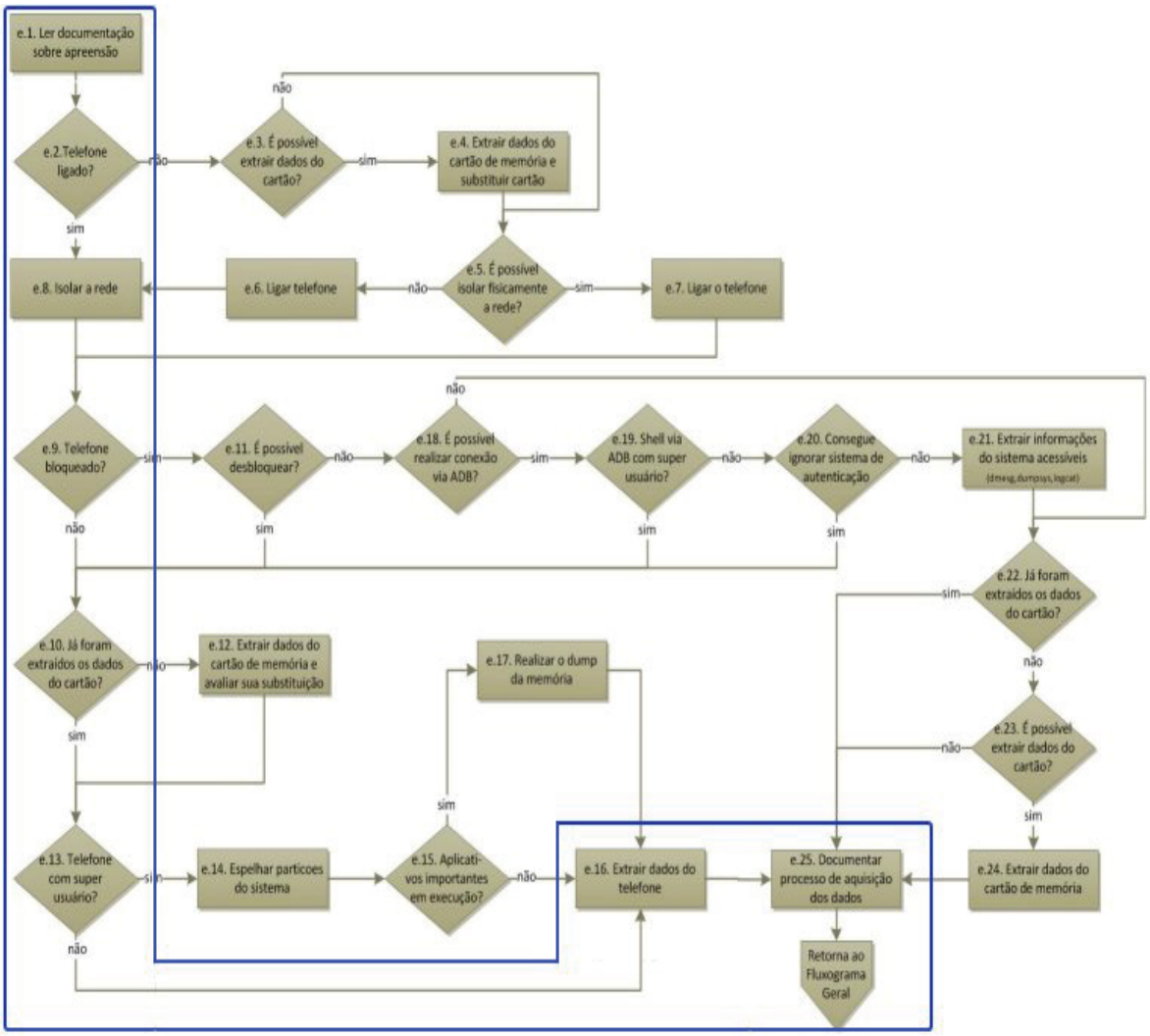


Figura 4.1: Exemplo da Adaptação 1

Na imagem 4.2, a introduzida uma alteração a metodologia anterior. Quando o dispositivo tem o mecanismo de segurança ativo, neste caso o ecrã bloqueado. Para contornar esse mecanismo é preciso instalar uma aplicação que tem por função desativar o mecanismo permitindo o acesso ao dispositivo.

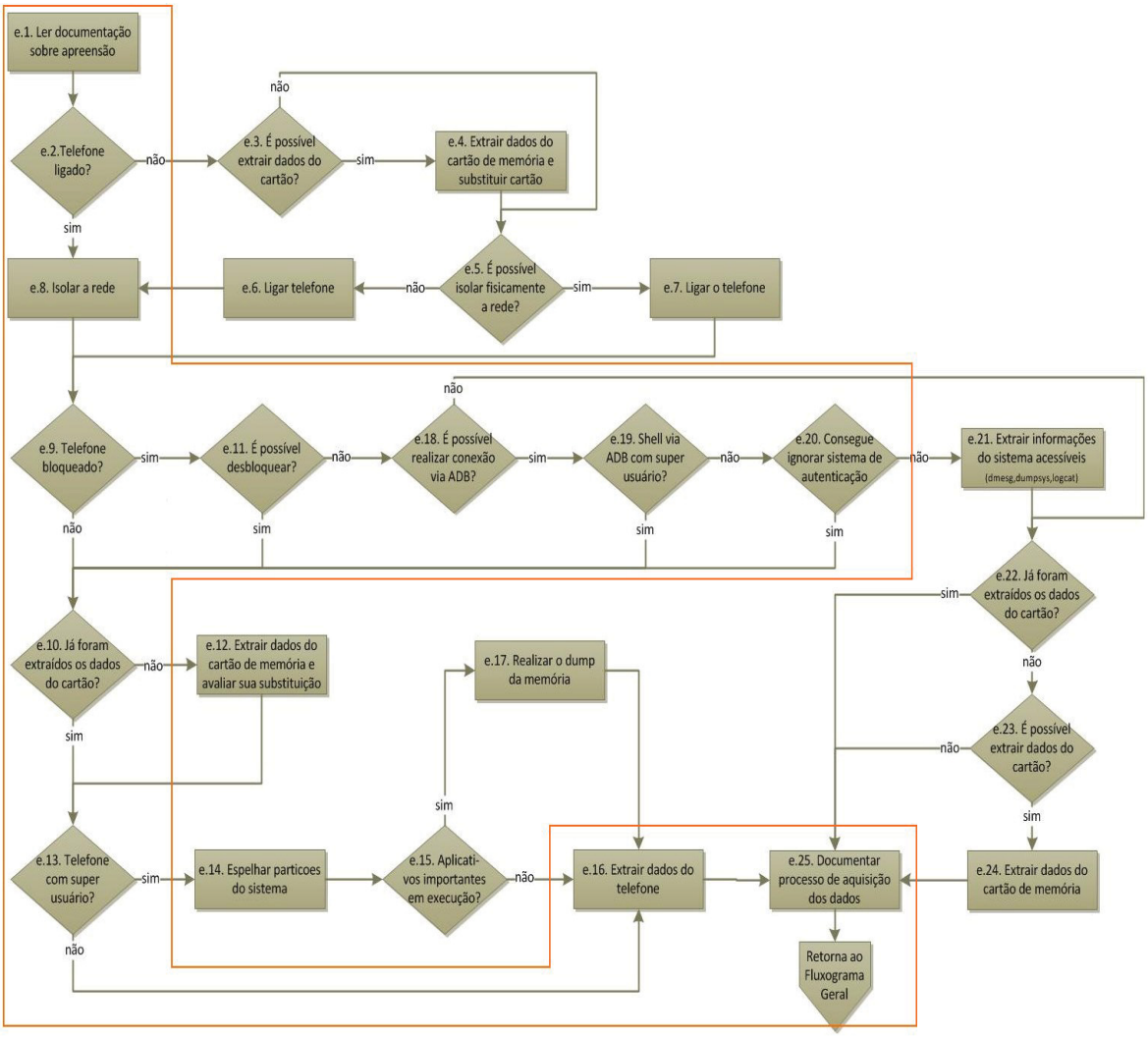


Figura 4.2: Exemplo da Adaptação 2

Capítulo 5

Implementação

5.1 Arquitetura do Sistema

5.1.1 Visão Geral

O modelo de analise forense definido anteriormente necessita de ter um sistema para extrair os dados do dispositivo, para esse processo foi implementado um sistema para obter essa informação.

Figura 5.1: Visão Geral do Sistema

O Sistema desenvolvido tem como objetivo principal, extrair informação do dispositivo independentemente de qual é a versão do SO *Android* instalado no equipamento. O sistema foi desenvolvido com base numa arquitetura Cliente-Servidor, com uma aplicação com funções de Servidor que envia os dados obtidos para outra aplicação com função de Cliente que gera um relatório com os dados recebidos.

5.1.2 Arquitetura Modular

A figura 5.2 ilustra a Arquitetura do Sistema. Cada aplicação tem como principio base de implementação a criação de um conjunto de módulos que interagem

entre si, permitindo ao sistema funcionar. A opção de se utilizar uma arquitetura modular apresenta vantagens, tanto a nível de manutenção, de implementação, de compreensão do sistema, mas a razão principal foi a facilidade de adicionar mais funcionalidades ás aplicações, sendo a primeira versão do sistema existem um conjunto de funcionalidades que terão de ser implementadas e tendo SO *Android* uma evolução muito rápida, isso implica que o sistema desenvolvido também se consiga adaptar as novas API do SO

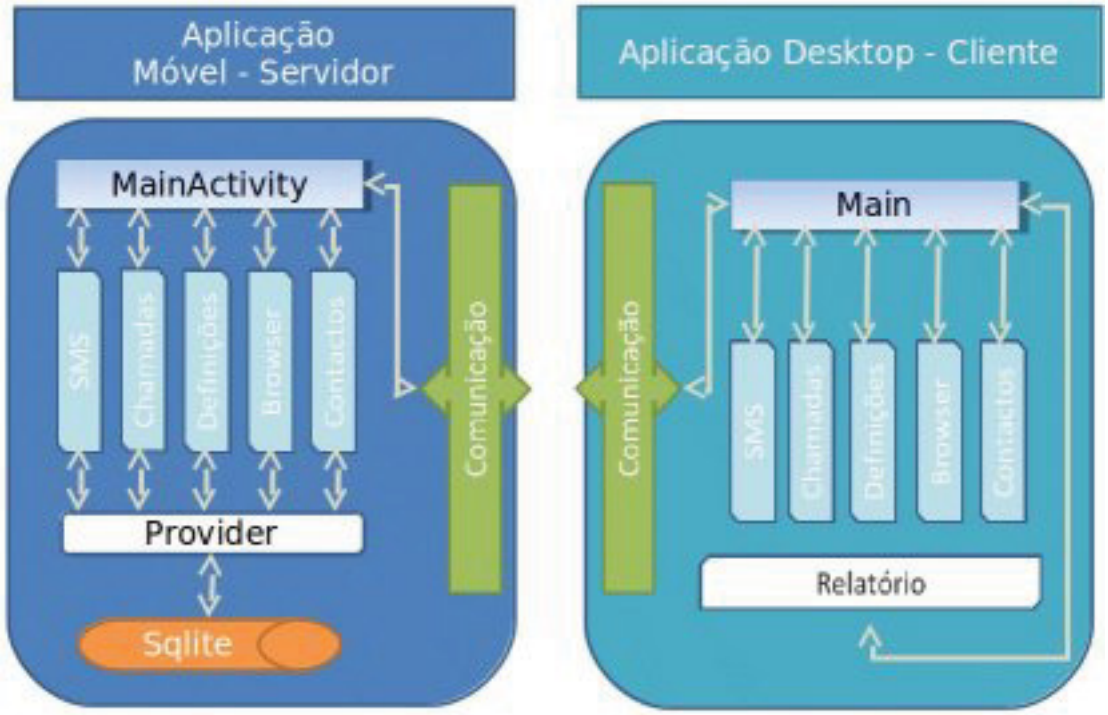


Figura 5.2: Arquitetura do Sistema

Cada aplicação tem um módulo de comunicação, que cria um canal de comunicação entre as aplicações e permite a transferência dos dados entre aplicação Servidor e a Aplicação Cliente.

A aplicação Servidor tem como base o desenvolvimento de aplicações para *Android*. São desenvolvidos 5 módulos, *SMS, Chamadas, Definições, Browser e Contactos*. Cada módulo através do *Provider* consegue aceder aos dados que pretende extrair da Base de Dados SQLite.

Na aplicação Cliente temos o módulo Main que controla a aplicação, fazendo a gestão dos dados que recebe pelo módulo de comunicação e produzindo um relatório com esses dados através do módulo Relatório.

Em cada aplicação os módulos que permitem manipular os dados estão replicadas

nas duas aplicações, de modo a fazer o tratamentos dos objetos que são enviados e recebidos.

5.2 Descriação das Classes e Métodos

5.2.1 Ferramentas e tecnologias utilizadas

Todo do sistema foi desenhado para funcionar num modelo de programação orientada a objetos, devido a restrição que existe para se desenvolver aplicações para o SO *Android* utilizou-se a linguagem JAVA, uma linguagem *Open Source*. Para o desenvolvimento da aplicação Servidor que consiste numa aplicação *Android*, utiliza-se o *Android* SDK que fornece as bibliotecas de API e as ferramentas necessárias para desenvolver, fazer testes e depurações as aplicações para o SO *Android*. Neste caso utilizou-se a API 19, por ser a ultima versão disponível é assim todo código ser aceitei em todas as versões do SO, com as devidas alterações para casa versão [56][56].

No ficheiro de configuração *AndrodiManifest.xml* foi removido a referência que existe a versão mínima e máxima da API, isto porque durante o desenvolvido da aplicação foi detetado um problema no funcionamento com a referência.

```
1 <uses-sdk android:minSdkVersion="5"
2 android:targetSdkVersion="19" />
```

Se este código estivesse no ficheiro de configuração a aplicação não funcionava em versões inferiores a SDK 8.

Para facilitar a configuração do sistema de desenvolvimento utiliza-se ADT (*Android Development Tools*) que é um plugin para o *Eclipse* IDE que é projetado para se ter ambiente de desenvolvimento integrado no qual se pode construir aplicativos *Android*. ADT amplia os recursos do Eclipse para que seja mais fácil criar novos projetos para *Android*, criar uma interface para a aplicação, adicionar pacotes com base na API *Android*, depurar as aplicações utilizando as ferramentas do SDK do *Android*, exportar a aplicação, entre outros [57]. Para a Aplicação Cliente utiliza-se também o JAVA como linguagem de desenvolvido, com a versão 7 do JRE (*Java Runtime Environment*) e a plataforma de desenvolvido *Eclipse*, neste caso com a versão 3.8.1, mas neste caso não existe restrições ao nível do editor.

O ADB é uma ferramenta incluída no SDK que disponibiliza uma interface para um emulador ou para um dispositivo *Android* que se conecta ao computador. O ADB tem por função permitir a comunicação entre o computador e o dispositivo móvel,

sendo um dos ponto cruciais em todo o sistema, garantir a ligação entre a aplicação Servidor e Cliente. Mas para que a execução de comando ADB seja possível através da conexão com o serviço disponível, a opção "Depuração USB" nas configuração do dispositivo tem de estar ativada, sendo esta uma restrição ao funcionamento de todo o sistema, devido a inviabilizar a comunicação entre as aplicações [58]. Para gerar os relatórios em forma PDF utiliza-se a API *OpenSource iText*, é uma biblioteca *OpenSource* com uma versão para JAVA. A sua utilização facilita a criação do relatório forense com base no dados que foram extraídos, otimizando todo o processo porque permite criar um relatório pré-formatado [59].

5.2.2 Diagrama de Classes

Na figura seguinte 5.3 podemos observar o modelo da estrutura da aplicação Servidor, onde se pode ver a interação entre as vários componentes da aplicação, temos um conjunto de módulos que fornecem serviços e dados a outras partes a aplicação. No anexo II.1 temos uma figura que ilustra o relacionamento entre as Classes da Aplicação Servidor, com base no Código da Aplicação.

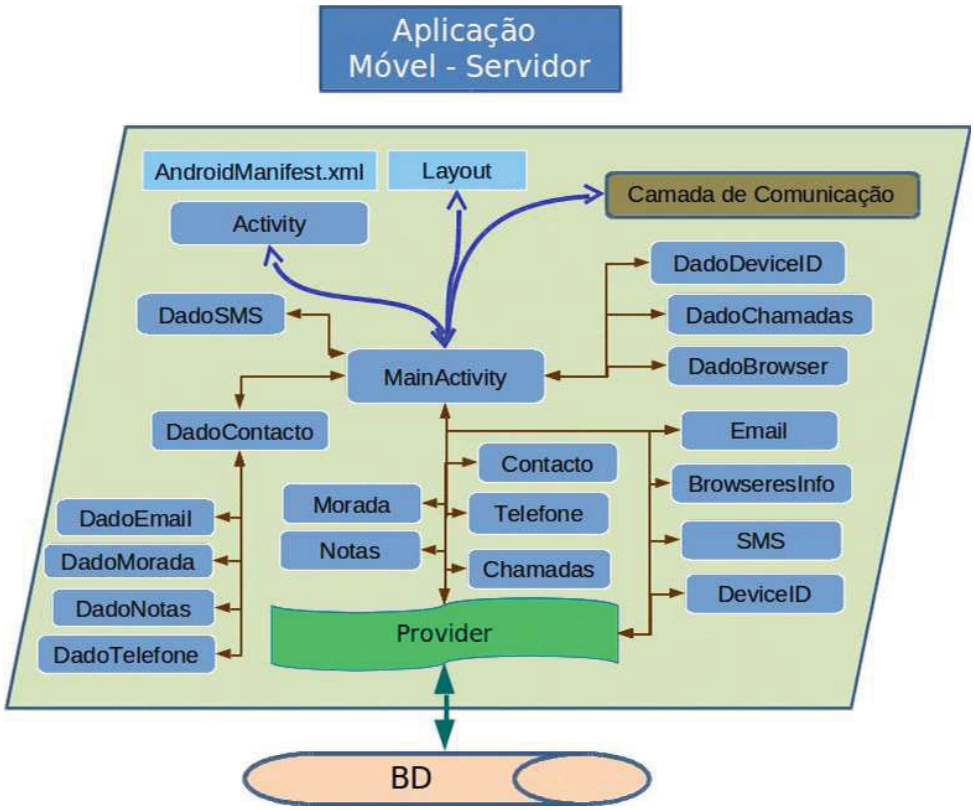


Figura 5.3: Relação Classes - Servidor

Os módulos *Contacto, Telefone, Morada, Notas, Email, Chamadas, SMS,Browser, DeviceID* vão carregar os dados solicitados pelo *Provider*. De referir que o módulo *Contactos* interage com os módulos *Telefone, Morada, Notas e Email*, sendo criado o objeto do tipo *Contactos*, com os dados desses módulo. Esta interação justifica-se porque um contacto que esteja registado na lista de contactos pode ter vários dados referentes a esses módulos.

O *AndroidManifest.xml* contem as permissões necessárias para que a aplicação consiga aceder a recursos do Sistema, todas aplicações *Android* precisam de ter este ficheiro com as respetivas permissões.

O módulo *MainActivity* é o responsável por gerir todos os processos da aplicação. O *Layout* é a interface da aplicação Servidor onde o utilizador irá selecionar na *check-Box* os dados a extrair.

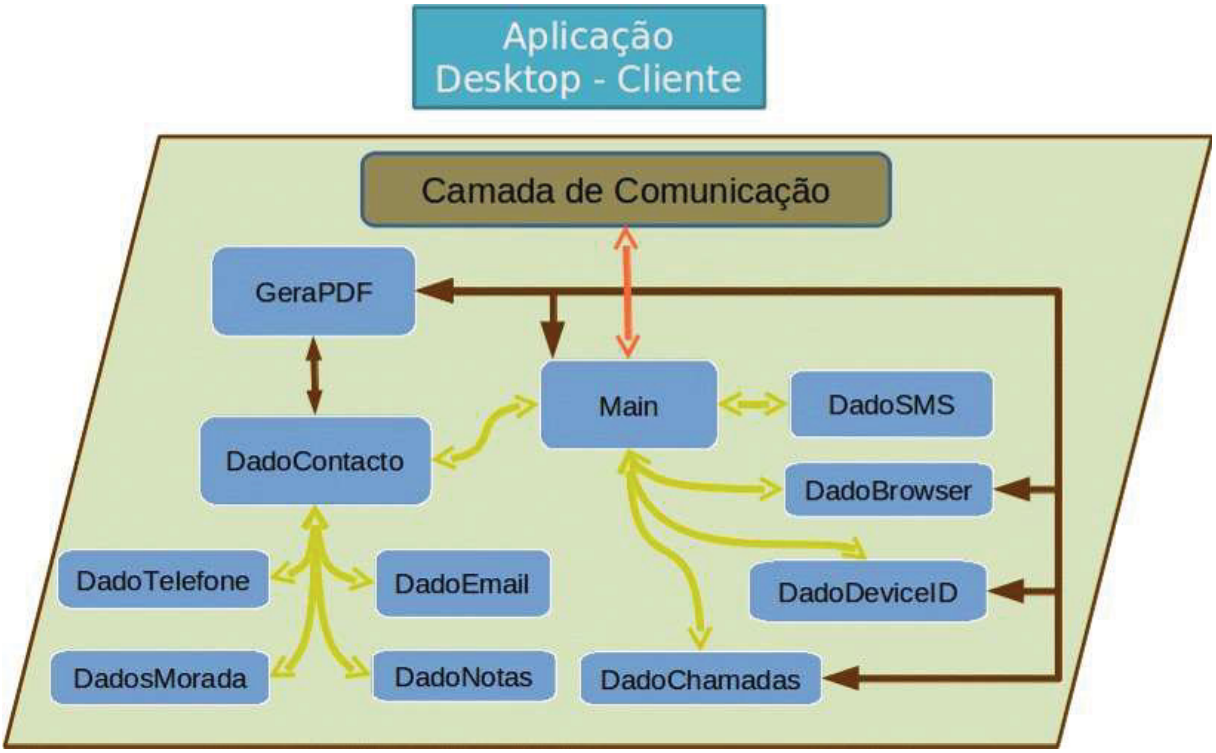


Figura 5.4: Relação Classes - Cliente

Na figura 5.4 podemos observar o modelo da estrutura da aplicação Cliente, sendo que a estrutura é muito semelhante a aplicação Servidor, principalmente na

forma como são definidas as Classes. Sendo que neste caso não existe interação com o *Provider*, sendo os dados processados de forma a serem enviados para a Classe GeraPDF responsável pela criação dos relatórios. No anexo III.1 temos uma figura que ilustra o relacionamento entre as Classes da Aplicação Cliente, com base no Código da Aplicação.

5.2.3 Classes Servidor

Para facilitar a descrição das Classes Servidor, só é feita referencia na listagem do código a algumas partes da Classe e dos métodos implementados.

Classe: ***DeviceID.Java***

Java.lang.Object
↬ *android.telephony.TelephonyManager*

A Classe *DeviceID* tem como função principal recolher sobre os serviços de comunicação e definições do dispositivo. Para se obter esta informações utiliza-se vários métodos da Classe *TelephonyManager* da API *Android*, de modo a criar um objeto da Classe *DadoDeviceID* é adicionar esse objeto a uma lista. Para se obter acesso a esta informação é necessário que sejam dadas permissões de acesso a informação para isso é necessário declara no ficheiro *AndroidManifest.xml*.

```
1 <uses-permission
2 android:name="android.permission.READ_PHONE_STATE" />
```

Listing 5.1: Parte do ficheiro *AndroidManifest.xml*

Para aceder aos dados é necessário referenciar a instância.
ctx.getSystemService(Context.TELEPHONY_SERVICE);

Para se obter os dados dos serviços e definições tem de ser obtidos na Classe *MainActivity* e depois enviados para a Classe *DeviceID*.

```
1 ...
2 TelephonyManager mngr = (TelephonyManager) getSystemService(
      Context.TELEPHONY_SERVICE);
3 tring id = mngr.getDeviceId();
4 String sv = mngr.getDeviceSoftwareVersion();
```

```
5  String simOperator = mngr.getSimOperatorName();
6  String simNumero = mngr.getSimSerialNumber();
7  String numero;
8  if (mngr.getLine1Number() != null) {
9    numero = "unavailable";
10 } else {
11   numero = (mngr.getLine1Number());
12 }
13 String nomeOperador = mngr.getNetworkOperatorName();
14 String simCode = mngr.getSimCountryIso();
15 String subId = mngr.getSubscriberId();
16 ...
```

Listing 5.2: Parte da Classe *MainActivity.java*

GetDeviceId() - Retorna o ID do dispositivo, por exemplo o IMEI para o GSM e o MEID (Mobile Equipment Identifier) ou ESN (*Electronic Signage Network*) para dispositivos CDMA.

GetDeviceSoftwareVersion() - Retorna o numero da versão do *software* para o dispositivo, o IMEI/SV para dispositivos GSM

GetSimOperatorName() -Retorna o SPN (*Service Provider Name*).

GetSimSerialNumber() - Retorna o numero de serie do SIM.

GetLine1Number() - Retorna o numero de telemóvel.

getNetworkOperatorName() - Retorna o nome do operador.

getSimCountryIso() - Retorna o ID do país.

getSubscriberId() - Retorna o ID da subscrição.

```
1  ...
2  objdevice.setDevice(ids);
3  objdevice.setDeviceSV(softwareVersion);
4  bjdevice.setNumeroTelefone(numeros);
5  objdevice.setSimOperator(simOperator);
6  objdevice.setSimNumero(simNumero);
7  objdevice.setNomeOperador(nomeOperador);
8  objdevice.setSimCode(simCode);
9  objdevice.subId(subId);
10 ...
11 lstdevice.add(objdevice);
12 ...
```

Listing 5.3: Parte da Classe *DeviceID.Java*

Classe: ***SMS.Java***

Java.lang.Object
↬ *InterfaceCursor*

A Classe *SMS* é responsável por extrair as SMS que estão no dispositivos, em todos os estados, recebidas, enviadas e rascunhos. A interface *Cursor* permite realizar consultas aos dados que estão na base de dados SQLlite, devolvendo os dados através dos métodos específicos. O *Cursor* tem de ser referenciado com a indicação do que se pretende consultar. Utiliza-se o *"content://sms/"* para referenciar o *Provider* URI do SDK.

Para se obter acesso a esta informação sobre as SMS é necessário que sejam dadas permissões de acesso a informação para isso é necessário declara no ficheiro *AndroidManifest.xml*.

```
1 <uses-permission
2 android:name="android.permission.READ_SMS" />
3 <uses-permission
4 android:name="android.permission.READ_CONTACTS" />
```

Listing 5.4: Parte do ficheiro *AndroidManifest.xml*

```
1  public List<DadoSMS> getAllSms() {
2    List<DadoSMS> lstSms = new ArrayList<DadoSMS>();
3    DadoSMS objSms = new DadoSMS();
4    Uri message = Uri.parse("content://sms/");
5    Cursor c = ctx.getContentResolver().query(message, null, null,
         null, null);
6    int totalSMS = c.getCount();
7    if (c.moveToFirst()){
8      for (int i = 0; i < totalSMS; i++) {
9      objSms = new DadoSMS();
10     objSms.setId(c.getString(c.getColumnIndexOrThrow"_id")));
11     objSms.setAddress(c.getString(c.getColumnIndexOrThrow("
           address")));
12     objSms.setMsg(c.getString(c.getColumnIndexOrThrow("body")));
13     objSms.setReadState(c.getString(c.getColumnIndex("read")));
14     objSms.setTime(c.getString(c.getColumnIndexOrThrow("date")));
15       if(c.getString( c.getColumnIndexOrThrow("type")).contains("
             1")){
16  objSms.setFolderName("inbox");
17       } else {
```

```
18  objSms.setFolderName("sent");
19        }
20        lstSms.add(objSms);
21  ...
```

Listing 5.5: Parte da Classe *SMS.Java*

GetColumnIndexOrThrow() - Retorna o valor com base no nome da coluna que é passada como parametro.

Classe: ***Chamadas.Java***

Java.lang.Object
↬ *android.provider.CallLog.Calls*

A Classe *Chamadas* é responsável por recolher os dados sobre chamadas que foram recebidas, efetuadas e perdidas, e modo a criar um objeto da Classe *Dado-Chamadas* é adicionar esse objeto a uma lista. Para recolher os dados utiliza-se o *Provider android.provider.CallLog.Calls*, que permite obter um conjunto importante de dados.

Para se obter acesso a esta informação sobre as Chamadas é necessário que sejam dadas permissões de acesso a informação para isso é necessário declara no ficheiro *AndroidManifest.xml*.

```
1 <uses-permission
2 android:name="android.permission.READ_CONTACTS" />
```

Listing 5.6: Parte do ficheiro *AndroidManifest.xml*

```
1  public List<DadoChamadas> getChamadas() {
2       List<DadoChamadas> lstchamadas = new  ArrayList<DadoChamadas
            >();
3       DadoChamadas objchamadas = new DadoChamadas();
4       String[] projection = new String[] { CallLog.Calls._ID,
            CallLog.Calls.CACHED_NAME,
5         CallLog.Calls.DATE, CallLog.Calls.DURATION,  CallLog.Calls.
              NUMBER, CallLog.Calls.TYPE };
6       Uri callUri = Uri.parse("content://call_log/calls");
7       Cursor c = ctx.getContentResolver().query(callUri, null,
8         null, null, null);
9       while (c.moveToNext()){
10            objchamadas = new Dado_Chamadas();
```

```
11          objchamadas.setID(c.getString(
12    c.getColumnIndex(android.provider.CallLog.Calls._ID)));
13          objchamadas.setNUMBER(c.getString(
14    c.getColumnIndex(android.provider.CallLog.Calls.NUMBER)));
15          SimpleDateFormat formatter = new SimpleDateFormat("dd-MMM
                -yyyy HH:mm");
16          String dateString = formatter.format(new Date(Long.
                parseLong(c.getString(c.etColumnIndex(android.provider
                .CallLog.Calls.DATE)))));
17  objchamadas.setDATE(dateString);
18  objchamadas.setCACHED_NAME(c.getString(
19    c.getColumnIndex(android.provider.CallLog.Calls.CACHED_NAME)))
        ;
20  objchamadas.setDURATION(c.getString(
21    c.getColumnIndex(android.provider.CallLog.Calls.DURATION)));
22  String type = c.getString(
23    c.getColumnIndex(android.provider.CallLog.Calls.TYPE));
24  if (type.equals("3")) {objchamadas.setTYPE("missed call");}
25  else if (type.equals("1")) {objchamadas.setTYPE("incoming call")
      ;}
26  else if (type.equals("2")) {objchamadas.setTYPE("outgoing call")
      ;}
27  lstchamadas.add(objchamadas);}
28     c.close();
29     return lstchamadas;
30 }
```

Listing 5.7: Parte da Classe *Chamadas.Java*

getColumnIndex(android.provider.CallLog.Calls.NUMBER))) - Numero da chamada.
getColumnIndex(android.provider.CallLog.Calls.DATE)))))
objchamadas.setDATE(dateString) - Data da chamada.
getColumnIndex(android.provider.CallLog.Calls.CACHED_NAME))) - Identificação se existir do contacto da chamada.
getColumnIndex(android.provider.CallLog.Calls.DURATION))) - Duração da Chamada.
getColumnIndex(android.provider.CallLog.Calls.TYPE)) - Tipo de Chamada, *incoming, outgoing or missed.*

Classe: ***BrowserInfo.Java***

Java.lang.Object
↬ *android.provider.Browser*

A Classe *BrowserInfo* é responsável pelo recolher os dados de acesso e de *Bookmark* e modo a criar um objeto da Classe *DadoBrowser* é adicionar esse objeto a uma lista.

Para recolher os dados utiliza-se o *Provider android.provider.Browser .BookmarkColumns*, que permite obter um conjunto importante de dados sobre a utilização do *Browser*.

Para se obter acesso a esta informação sobre as o *Browser* é necessário que sejam dadas permissões de acesso a informação para isso é necessário declara no ficheiro *AndroidManifest*.

```
<uses-permission
android:name="com.android.browser.permission.
    READ_HISTORY_BOOKMARKS" />
```

Listing 5.8: Parte do ficheiro *AndroidManifest.xml*

```
public List<DadoBrowser> getAllBookmark() {
  List<DadoBrowser> lstBrowser = new ArrayList<Dado_Browser>();
  DadoBrowser objBrowser = new DadoBrowser();
    Cursor c = ctx.getContentResolver().query(android.provider.
        Browser.BOOKMARKS_URI, proj, null, null, null);
  c.moveToFirst();
  int titleIdx = c.getColumnIndex(Browser.BookmarkColumns.TITLE);
  int urlIdx = c.getColumnIndex(Browser.BookmarkColumns.URL);
  int visitsIdx = c.getColumnIndex(Browser.BookmarkColumns.VISITS
      );
  int dataIdx = c.getColumnIndex(Browser.BookmarkColumns.DATE);
  while (c.isAfterLast() == false){
    objBrowser = new Dado_Browser();
    objBrowser.setTitle(c.getString(titleIdx));
    objBrowser.setURL(c.getString(urlIdx));
    SimpleDateFormat formatter = new SimpleDateFormat("dd-MMM-
        yyyy HH:mm");
    String dateString = formatter.format(new Date(Long.parseLong(
        c.getString(dataIdx))));
    objBrowser.setData(dateString);
    objBrowser.setAcessos(c.getString(visitsIdx));
```

```
18        lstBrowser.add(objBrowser);
19      c.moveToNext();
20      }
21      return lstBrowser;
22  }
```

Listing 5.9: Parte da Classe *BrowserInfo.Java*

c.getColumnIndex(Browser.BookmarkColumns.TITLE) - Titulo do site
c.getColumnIndex(Browser.BookmarkColumns.URL) - URL (*Uniform Resource Locator*) do site
c.getColumnIndex(Browser.BookmarkColumns.VISITS) - Numero de acesso
c.getColumnIndex(Browser.BookmarkColumns.DATE) - Ultima data de acesso

Classe: ***Email.Java***

Java.lang.Object
↳ *android.provider.ContactsContract.CommonDataKinds.Email*

A Classe *Email* é responsável por recolher os dados referentes aos *emails* do contacto e criar um objeto da Classe *DadoEmail* é adicionar esse objeto a uma lista. Para recolher os dados utiliza-se o *Provider android.provider.ContactsContract.Data*, que permite obter os dados referentes aos *email*.

Para se obter acesso a esta informação sobre os email dos contactos é necessário que sejam dadas permissões de acesso a informação para isso é necessário declara no ficheiro *AndroidManifest.xml*.

```
1 <uses-permission
2 android:name="android.permission.READ_CONTACTS" />
```

Listing 5.10: Parte do ficheiro *AndroidManifest.xml*

```
1  public List<DadoEmail> getEmail() {
2    Cursor emailCursor = this.ctx.getContentResolver().query(
3    ContactsContract.CommonDataKinds.Email.CONTENT_URI, projection,
4    ContactsContract.CommonDataKinds.Email.CONTACT_ID + " = ?"new
         String[] { _IDContato }, null);
5    int IndexEmail;
6    List<DadoEmail> Email = new ArrayList<DadoEmail>();
7    while (emailCursor.moveToNext()) {
8      DadoEmail E = new DadoEmail();
```

```
9       IndexEmail = emailCursor.
10      getColumnIndex(ContactsContract.CommonDataKinds.Email.DATA);
11      E.setEmail(emailCursor.getString(IndexEmail));
12      Email.add(E);}
13     emailCursor.close();
14     return Email;
15   }
```

Listing 5.11: Parte da Classe *Email.Java*

getColumnIndex(ContactsContract.CommonDataKinds.Email .DATA) - Constantes para a tabela de dados, que contém os dados ligados a um contacto.

Classe: ***Morada.Java***

Java.lang.Object
↬ *android.provider.ContactsContract.CommonDataKinds.StructuredPostal*

A Classe *Morada* é responsável por recolher os dados referentes aos moradas do contacto e criar um objeto da Classe *DadoMorada* é adicionar esse objeto a uma lista. Para recolher os dados utiliza-se o *Provider android.provider.ContactsContract .CommonDataKinds.StructuredPostal*, que permite obter os dados referentes a morada.

Para se obter acesso a esta informação sobre as moradas dos contactos é necessário que sejam dadas permissões de acesso a informação para isso é necessário declara no ficheiro *AndroidManifest.xml*.

```
1 <uses-permission
2 android:name="android.permission.READ_CONTACTS" />
```

Listing 5.12: Parte do ficheiro *AndroidManifest.xml*

```
1 public List<DadoMorada> getMorada() {
2   List<DadoMorada> enderecos = new ArrayList<DadoMorada>();
3   final String[] projection = {
4     ContactsContract.CommonDataKinds.StructuredPostal.STREET,
5     ContactsContract.CommonDataKinds.StructuredPostal.CITY,
6     ContactsContract.CommonDataKinds.StructuredPostal.REGION,
7     ContactsContract.CommonDataKinds.StructuredPostal.POSTCODE,
8     ContactsContract.CommonDataKinds.StructuredPostal.COUNTRY,
9     ContactsContract.CommonDataKinds.StructuredPostal.TYPE };
```

```
10  String[] whereParameters = new String[] { _IDContato,
11  ContactsContract.CommonDataKinds.StructuredPostal.
        CONTENT_ITEM_TYPE };Cursor moradaCursor = this.ctx.
        getContentResolver().query(
12  ContactsContract.Data.CONTENT_URI, projection,
13  ContactsContract.Data.CONTACT_ID + " = ? AND "+
        ContactsContract.Data.MIMETYPE + " = ?",whereParameters,
        null);
14  while (moradaCursor.moveToNext()) {
15    Dado_Morada morada = new Dado_Morada();
16    morada.Rua = moradaCursor.getString(moradaCursor.
          getColumnIndex(ContactsContract.CommonDataKinds.
          StructuredPostal.STREET));
17    morada.Cidade = moradaCursor.getString(moradaCursor.
          getColumnIndex(ContactsContract.CommonDataKinds.
          StructuredPostal.CITY));
18    morada.Regiao = moradaCursor.getString(moradaCursor.
          getColumnIndex(ContactsContract.CommonDataKinds.
          StructuredPostal.REGION));
19    morada.Codigo = moradaCursor.getString(moradaCursor.
          etColumnIndex(ContactsContract.CommonDataKinds.
          StructuredPostal.POSTCODE));
20    morada.Pais = moradaCursor.getString(moradaCursor.
          getColumnIndex(ContactsContract.CommonDataKinds.
          StructuredPostal.COUNTRY));
21    morada.Tipo = moradaCursor.getString(moradaCursor.
          getColumnIndex(ContactsContract.CommonDataKinds.
          StructuredPostal.TYPE));
22    enderecos.add(morada);}
23  moradaCursor.close();
24  return enderecos;}
```

Listing 5.13: Parte da Classe *Morada.Java*

ContactsContract.CommonDataKinds.StructuredPostal.STREET)) - Rua
ContactsContract.CommonDataKinds.StructuredPostal.CITY)) - Cidade
ContactsContract.CommonDataKinds.StructuredPostal.REGION)) - Região
ContactsContract.CommonDataKinds.StructuredPostal.COUNTRY)) - Pais
ContactsContract.CommonDataKinds.StructuredPostal.TYPE)) - Tipo Morada
ContactsContract.CommonDataKinds.StructuredPostal.POSTCODE)) - Codigo Postal

Classe: ***Notas.Java***

Java.lang.Object
↳ *android.provider.ContactsContract.CommonDataKinds.Note*

A Classe *Nota* é responsável por recolher os dados referentes as notas do contacto e criar um objeto da Classe *DadoNotas* é adicionar esse objeto a uma lista. Para recolher os dados utiliza-se o *Provider android.provider.ContactsContract .CommonDataKinds.Note*, que permite obter os dados referentes as notas. Para se obter acesso a esta informação sobre as notas referentes aos contactos é necessário que sejam dadas permissões de acesso a informação para isso é necessário declara no ficheiro *AndroidManifest.xml*.

```
1 <uses-permission
2 android:name="android.permission.READ_CONTACTS" />
```

Listing 5.14: Parte do ficheiro *AndroidManifest.xml*

```
1  public List<DadoNotas> getNotas() {
2    List<DadoNotas> notas = new ArrayList<DadoNotas>();
3    String where = ContactsContract.Data.CONTACT_ID + " = ? AND "
4    + ContactsContract.Data.MIMETYPE + " = ?";
5    String[] whereParameters = new String[] { _IDContato,
6    ContactsContract.CommonDataKinds.Note.CONTENT_ITEM_TYPE };
7    Cursor notaCursor = this.ctx.getContentResolver().query(
8    ContactsContract.Data.CONTENT_URI, null, where,whereParameters,
         null);
9    while (notaCursor.moveToNext()) {
10     Dado_Notas nota = new Dado_Notas();
11     nota.Notas = notaCursor.getString(notaCursor.getColumnIndex(
           ContactsContract.CommonDataKinds.Note.NOTE));
12     notas.add(nota);}
13   notaCursor.close();
14   return notas;}
```

Listing 5.15: Parte da Classe *Notas.Java*

ContactsContract.CommonDataKinds.Note.NOTE - Nota

Classe: ***Telefone.Java***

Java.lang.Object

↬ *android.provider.ContactsContract.CommonDataKinds.Phone*

A Classe *Telefone* é responsável por recolher os dados referentes números de telefone do contacto e criar um objeto da Classe *DadoTelefone* é adicionar esse objeto a uma lista. Para recolher os dados utiliza-se o *Provider android.provider.ContactsContract .CommonDataKinds.Phone*, que permite obter os dados referentes as notas. Para se obter acesso a esta informação sobre os numero de telefone do contacto é necessário que sejam dadas permissões de acesso a informação para isso é necessário declara no ficheiro *AndroidManifest.xml*.

```
1 <uses-permission
2 android:name="android.permission.READ_CONTACTS" />
```

Listing 5.16: Parte do ficheiro *AndroidManifest.xml*

```
1  public List<DadoTelefone> getTelefones() {
2    Cursor C_Telefones = this._ctx.getContentResolver().query(
3    ContactsContract.CommonDataKinds.Phone.CONTENT_URI, null,
4    ContactsContract.CommonDataKinds.Phone.CONTACT_ID + " = "+
         _IDContato,
5    null, null);
6    int IndexTelefone;
7    List<DadoTelefone> Telefones = new ArrayList<DadoTelefone>();
8    while (C_Telefones.moveToNext()) {
9      DadoTelefone Telefone = new DadoTelefone();IndexTelefone =
           C_Telefones.getColumnIndex(ContactsContract.
           CommonDataKinds.Phone.NUMBER);
10     Telefone.setTelefone(C_Telefones.getString(IndexTelefone));
11     Telefones.add(Telefone);}
12   C_Telefones.close();
13   return Telefones;
14 }
```

Listing 5.17: Parte da Classe *Telefone.Java*

C_Telefones.getColumnIndex(ContactsContract.CommonDataKinds .Phone.NUMBER) - Numero de Telefone

Classe: ***Comunicacao.Java***

A Classe *Comunicacao* permite a comunicação entre as aplicações. Implementa o *ServerSocket* que permite estabelecer a ligação com a aplicação cliente, tem im-

plementados varios métodos cada um com uma função específica, dos quais destaco.
iniciaComunicacao(): Estabelece a ligação com a aplicação Cliente.
canalSaida(List obj): Permite enviar os objetos para aplicação Cliente.

```
1  ...
2  public ServerSocket iniciaComunicacao(){
3    try {
4      server = new ServerSocket(59900);
5    }
6  ...
7  public void canalSaida(List obj) {
8    try {
9      cSaida.writeObject(obj);
10     cSaida.flush();
11   }
12 ...
```

Listing 5.18: Parte da Classe *Comunicacao.Java*

Classe: ***MainActivity.Java***

Java.lang.Object

↳ *android.content.Context*
↳ *android.content.ContextWrapper*
↳ *android.view.ContextThemeWrapper*
↳ *android.app.Activity*

A Classe *MainActivity* é a classe responsável por gerir todo o sistema, é a componente chave de todo sistema, mas para que exista interação com a interface do utilizador ela vai herdar propriedades da Classe *Activity*. Na construção da Classe é necessário implementar alguns métodos que vão permitir a interação dos eventos gerados na classe. Método *onCreate(Bundle)* que vai inicializar toda a atividade da Classe, toda as atividade são declaradas neste métodos.

Tem de ser alterada falta a classe comunicação

```
1 protected void onCreate(Bundle savedInstanceState) {
2     super.onCreate(savedInstanceState);
3     setContentView(R.layout.activity_main);
4     buttonSend = (Button) findViewById(R.id.button2);
5     buttonClose = (Button) findViewById(R.id.button3);
```

```
6       ...
7       ublic void onClick(View arg0) {
8         try {
9   ...
10  //Botão enviar dados
11  canalEntrada = new ObjectInputStream(socket.getInputStream());
12  canalSaida = new ObjectOutputStream(socket.getOutputStream());
13  String s = (String) canalEntrada.readObject();
14  ((TextView) MainActivity.this.findViewById(R.id.txtMsg)).setText
        (s);
15
16  if (!ListaContatos.isEmpty()) {
17    canalSaida.writeObject(ListaContatos);
18    canalSaida.flush();}
19    listaContatos.clear();}
20  ...
21
22  // checkBox de seleção de dados
23  public void addCheckbox() {
24    checkBox_contacto = (CheckBox)findViewById(R.id.
          checkBox_contato);
25    checkBox_sms = (CheckBox) findViewById(R.id.checkBox_sms);
26    ...
27    checkBox_contato.setOnClickListener(new OnClickListener() {
28      public void onClick(View v) {
29      if (((CheckBox) v).isChecked()Toast.makeText(
            getApplicationContext(), "Contactos!!",  Toast.
            LENGTH_LONG).show();
30        ListaContatos = conn.getContactos();}}
31      });
32
33      ...
```

Listing 5.19: Parte da Classe *MainActivity.Java*

Uma atividade é uma interação que o utilizador pode fazer. De modo que a classe *Activity* cria uma janela onde vai ser colocadas a interface do utilizador através do setContentView. Existem um método que é implementado herdado da Classe de *Activity:onCreate (Bundle)* onde é inicializada a atividade. Onde se implementar o *setContentView(int)* com um recurso de *layout* de definir sua interface do utilizador e utilizando o metodo *findViewById(int)* para recuperar os *widgets* que permitem a interação com a aplicação.

AndroidManifest.xml

A tarefa do *AndroidManifest* é de mapear as *activitys* e todo projeto android deve ter um arquivo *AndroidManifest.xml*, que descreve os componentes de aplicação, define nomes para as *activitys*, modos de orientação do ecrã (vertical, horizontal ou ambos), declara permissões para ser possivel aceder à recursos como o GPS ou *Internet* por exemplo, lista também as bibliotecas que a aplicação vão usar e qual *activity* iniciará primeiro quando a aplicação iniciar.

```
1  ...
2 <uses-permission
3     android:name="android.permission.INTERNET" />
4 <uses-permission
5     android:name="android.permission.ACCESS_NETWORK_STATE"  />
6 <uses-permission
7     android:name="android.permission.READ_CONTACTS"  />
8 <uses-permission
9     android:name="android.permission.ACCESS_WIFI_STATE" />
10 <uses-permission
11    android:name="android.permission.READ_SMS" />
12 <uses-permission
13    android:name="android.permission.READ_CALL_LOG" />
14 <uses-permission
15    android:name="android.permission.READ_PHONE_STATE" />
16 <uses-permission
17    android:name="android.permission.CALL_PHONE"  />
18 <uses-permission
19    android:name="com.android.browser.permission.
          READ_HISTORY_BOOKMARKS" />
20 ...
```

Listing 5.20: Parte do ficheiro *AndroidManifest.xml*

Classe: ***DadoBrowser.java***

Esta classe permite a criação de um objeto do tipo *DadoBrowser*. Tendo os seguintes atributos do tipo *String titulos, url, acessos, dataAcesso*. Todos este atributo são acedido por métodos, os quais podem alterar ou aceder o dado do atributo. O atributo *serialVersionUID* permite que se consiga recuperar o objeto que foi envia pelo canal criado pelo *ObjectOutputStream*.

```
1 public class DadoBrowser implements Serializable {
2   private static final long serialVersionUID = 1L;
3   private String titulos, url, acessos, dataAcesso;
4 ...
5   public String getTitle(){ return titulos; }
6   public void setTitle(String titulo) { this.titulos = titulo;}
7 ...
8 }
```

Listing 5.21: Parte da Classe *DadoBrowser.java*

Classe: ***DadoChamadas.java***

Esta classe permite a criação de um objeto do tipo *DadoChamadas*. Tendo os seguintes atributos do tipo *String ids, nomes, datas, duracoes, numeros, tipos*. Todos este atributo são acedido por métodos, os quais podem alterar ou aceder o dado do atributo. O atributo *serialVersionUID* permite que se consiga recuperar o objeto que foi envia pelo canal criado pelo *ObjectOutputStream*.

```
1 public class Dado_Chamadas implements Serializable {
2   private static final long serialVersionUID = 1L;
3   private String ids, nomes, datas, duracoes, numeros, tipos;
4 ...
5   public String getID(){  return ids; }
6   public void setID(String id) {this.ids = id; }
7 ...
8 }
```

Listing 5.22: Parte da Classe *DadoChamadas.Java*

Classe: ***DadoContacto.java***

Esta classe permite a criação de um objeto do tipo *DadoContacto*. Tendo os seguintes atributos do tipo *String ids, nomes*. Existe nesta classe quatro atributos do tipo *List* que se referem a lista de obejcto das Classe *DadoEmail, DadoTelefone, DadoMorada, DadoNotas*. Todos estes atributos são acedido por métodos, os quais podem alterar ou aceder o dado do atributo. O atributo *serialVersionUID* permite que se consiga recuperar o objeto que foi envia pelo canal criado pelo *ObjectOutputStream*.

```
1 public class DadoContacto implements Serializable {
2   private String ids, nomes;
```

```
3   private List<DadoEmail> emails;
4   private List<Dadotelefone> telefones;
5   private List<DadoMorada> moradas;
6   private List<DadoNotas> notas;
7   private static final long serialVersionUID = 1L;
8 ..
9   public String getID(){ return ids; }
10  public void setID(String id){ this.ids = id; }
11 ...
```

Listing 5.23: Parte da Classe *DadoContacto.java*

Classe: ***DadoDeviceID.java***

Esta classe permite a criação de um objeto do tipo *DadoDeviceID*. Tendo o seguinte atributos do tipo *String ids, idsv, simOperator, serialNumero, numeroTelefone,nomeOperador,simCode, subId*. Todos este atributo são acedido por métodos, os quais podem alterar ou aceder o dado do atributo. O atributo *serialVersionUID* permite que se consiga recuperar o objeto que foi envia pelo canal criado pelo *ObjectOutputStream*.

```
1 ublic class DadoDeviceID implements Serializable {
2  private static final long serialVersionUID = 1L;
3       private String ids, idsv, simOperator, serialNumero,
4       numeroTelefone,nomeOperador,simCode, subId;
5 ...
6       public String getDevice(){ return ids;}
7       public void setDevice(String id){ this.ids = id; }
8 ...
```

Listing 5.24: Parte da Classe *DadoDeviceID.java*

Classe: ***DadoEmail.java***

Esta classe permite a criação de um objeto do tipo *DadoEmail*. Tendo o seguinte atributo do tipo *String emails*. Todos este atributo são acedido por métodos, os quais podem alterar ou aceder o dado do atributo. O atributo *serialVersionUID* permite que se consiga recuperar o objeto que foi envia pelo canal criado pelo *ObjectOutputStream*.

```
1 public class DadoEmail implements Serializable {
2       private static final long serialVersionUID = 1L;
```

```
3       private String emails;
4 ...
5       public String getEmail(){ return email;}
6       public void setEmail(String email){ emails = email;}
7 ...
```

Listing 5.25: Parte da Classe *DadoEmail.java*

Classe: ***DadoMorada.java***

Esta classe permite a criação de um objeto do tipo *DadoMorada*. Tendo os seguintes atributos do tipo *String ruas, cidades, regioes, codigos, paises, tipos*. Todos este atributo são acedido por métodos, os quais podem alterar ou aceder o dado do atributo. O atributo *serialVersionUID* permite que se consiga recuperar o objeto que foi envia pelo canal criado pelo *ObjectOutputStream*.

```
1 public class DadoMorada implements Serializable {
2       private static final long serialVersionUID = 1L;
3       public String ruas, cidades, regioes, codigos, paises, tipos
            ;
4 ...
5       public void setTipo(String tipo){ tipos = tipo; }
6       public String getTipo(){ return tipos; }
7 ...
```

Listing 5.26: Parte da Classe *DadoMorada.java*

Classe: ***DadoNotas.java***

Esta classe permite a criação de um objeto do tipo *DadoNotas*. Tendo o seguinte atributo do tipo *String notas*. Todos este atributo são acedido por métodos, os quais podem alterar ou aceder o dado do atributo. O atributo *serialVersionUID* permite que se consiga recuperar o objeto que foi envia pelo canal criado pelo *ObjectOutputStream*.

```
1 public class DadoNotas implements Serializable {
2     private static final long serialVersionUID = 1L;
3     public String notas;
4 ...
5     public String getNotas(){ return notas;}
6     public void setNotas(String nota){ notas = nota; }
7 ...
```

Listing 5.27: Parte da *DadoNotas.java*

Classe: ***DadoSMS.java***

Esta classe permite a criação de um objeto do tipo textitDadoSMS. Tendo o seguinte atributo do tipo *String ids, numeroSMS, mensagem, estado, tempo, tipo.* Todos este atributo são acedido por métodos, os quais podem alterar ou aceder o dado do atributo. O atributo *serialVersionUID* permite que se consiga recuperar o objeto que foi envia pelo canal criado pelo *ObjectOutputStream.*

```
1 public class DadoSMS implements Serializable {
2     private String ids, numeroSMS, mensagem, estado, tempo, tipo;
3    private static final long serialVersionUID = 1L;
4 ...
5    public String getId() { return ids; }
6    public void setId(String id) { this.ids = id; }
7 ...
```

Listing 5.28: Parte da Classe *DadoSMS.java*

Classe: ***DadoTelefone.java***

Esta classe permite a criação de um objeto do tipo *DadoTelefone.* Tendo o seguinte atributo do tipo *String telefones.* Todos este atributo são acedido por métodos, os quais podem alterar ou aceder o dado do atributo. O atributo *serialVersionUID* permite que se consiga recuperar o objeto que foi envia pelo canal criado pelo *ObjectOutputStream.*

```
1 public class DadoTelefone implements Serializable {
2     private static final long serialVersionUID = 1L;
3     private String telefones;
4 ...
5     public String getTelefone() { return telefone; }
6     public void setTelefone(String telefone) { telefones =
          telefone; }
7 ...
```

Listing 5.29: Parte da Classe *DadoTelefone.java*

5.2.4 Classes Cliente

Para facilitar a descrição das Classes Cliente, só é feita referencia na listagem do código a algumas partes da Classe e dos métodos implementados. Não é feita referencia as seguintes Classes *DadoTelefone, DadoSMS, DadoContacto, DadoDeviceID , DadoEmail, DadoMorada, DadoNotas, DadoBrowser, DadoChamadas*porque são idênticas as que foram descritas nas Classes Servidor

Classe: ***main.java***

A Classe *Main* é a classe responsável por gerir todo o sistema na aplicação Cliente. Tem como função principal tratar os objetos recebidos é encaminha-los para o Classe GeraPDF.

```
1  public static void main(String[] args) throws IOException,
2  ClassNotFoundException {
3  ...
4  readObject = new ArrayList<Dadocontacto>();
5  ...
6  obj = (List) comunica.canalEntrada();
7  while (true) {
8       obj = (ArrayList) canalEntrada.readObject();
9       ListIterator litr = obj.listIterator();
10      if (litr.next().getClass().getSimpleName().equals("
            Dadocontacto")) {
11           readObject = (ArrayList<DadoCcontacto>) obj;
12           for (int i = 0; i < readObject.size(); i++) {
13              DadoContacto v = (DadoContacto) readObject.get(i);
14              lista.add(v);
15            }
16            litr.remove();
17        }
18 ...
19 GeraPDF.ficheiro_xml(lista,listaobjectSMS,listaobjectChamadas,
       listaobjectBrowser,listaobjectDeviceID);
20 ...
```

Listing 5.30: Parte da Classe *main.java*

Classe: ***Comunicação.Java***

A Classe *Comunicacao* permite a comunicação entre as aplicações. Implementa o *Socket* que permite estabelecer a ligação com a aplicação Servidor, tem imple-

mentados varios métodos cada um com uma função específica, dos quais destaco.
iniciaComunicacao(): Estabelece a ligação com a aplicação Servidor.
objectInput(): Permite receber os objetos enviados pela aplicação Servidor.
fechaCanalEntrada(): Encerra a ligação com aplicação Servidor.

```
1  public Socket iniciaComunicacao(){
2    try {
3      socket = new Socket("127.0.0.1", 59900);
4      System.out.println("Criou socket...\n");
5  }
6  ...
7  public void objectInput(){
8    try {
9      cEntrada = new ObjectInputStream(socket.getInputStream());
10   }
11 ...
12 public Object canalEntrada() throws ClassNotFoundException,
       IOException{
13   List obj2 = new ArrayList<Object>();
14   obj2 = (ArrayList) cEntrada.readObject();
15   if( obj2.size() == 0){
16     System.out.println("\nobject 2-  " + obj2.size());
17     fechaComunicacao();
18 }
19 ...
20 public void fechaCanalEntrada(){
21   try {
22     cEntrada.close();
23 }
24 ...
```

Listing 5.31: Parte da Classe *Comunicacao.Java*

Classe: ***GeraPDF.java***

A Classe *GeraPDF* gere o processo final da aplicação Cliente, criar um relatório com base nos dados recebidos. Para isso utiliza uma API *OpenSource iText*, que permite a criação de documentos em formato PDF. Como métodos principais da Classe temos o *addConteudo, criaTabelaBrowser, criaTabelaChamadas, criaTabelaContacto, criaTabelaDefinicoes, criaTabelaSMS*. O método *addConteudo* tem por função adicionar as listas de objetos ao métodos correspondentes, como por exemplo a *listaobjectSMS* ao método *criaTabelaSMS*.

```
1 private static void addConteudo(Document document,ArrayList<
    Dado_contacto>lista,
2 ArrayList<Dado_SMS> listaobjectSMS,
3 ArrayList<Dado_Chamadas> listaobjectChamadas,
4 ArrayList<Dado_Browser> listaobjectBrowser,
5 ArrayList<Dado_DeviceID> listaobjectDeviceID,
6 PdfWriter writer) throws DocumentException {
7 ...
8 criaTabelaContacto(subCatPart, lista);
9 addEmptyLine(subPara,1);
10 subPara = new Paragraph("SMS no Dispositivo", subFont);
11 subCatPart = catPart.addSection(subPara);
12 ...
13 criaTabelaSMS(subCatPart,listaobjectSMS);
14 ...
15 criaTabelaChamadas(subCatPart,listaobjectChamadas);
16 ...
17 criaTabelaBrowser(subCatPart,listaobjectBrowser);
18 ...
19 criaTabelaDefinicoes(subCatPart,listaobjectDeviceID);
20 ...
21 document.add(catPart);
22 ...
```

Listing 5.32: Parte da Classe *GeraPDF.java*

O método *criaTabelaSMS* vai criar uma tabela com os dados existirem na lista de SMS, isso se existirem dado, caso contrario cria uma tabela com a indicação de não existirem dados. O método esta preparado para para verificar a lista de objeto e extrair os dados criando uma tabela formatada de forma a facilitar a leitura dos dados. Os restantes métodos *criaTabelaBrowser, criaTabelaChamadas, criaTabelaContacto, criaTabelaDefinicoes*, tem um modo de funcionamento idêntico.

```
1 private static void criaTabelaSMS(Section subCatPartSMS,
2 ArrayList<Dado_SMS> listaobjectSMS)throws DocumentException {
3 ...
4   if(listaobjectSMS.isEmpty()){
5     c2 = new PdfPCell(new Phrase("Não contém informação"));
6     c2.setHorizontalAlignment(Element.ALIGN_CENTER);
7     c2.setColspan(4);
8     tableSMS.addCell(c2);
9 ...
```

```
10    subCatPartSMS.add(tableSMS);
11    } else{
12    for(int p = 0; p<listaobjectSMS.size();p++ ) {
13      osms = (Dado_SMS) listaobjectSMS.get(p);
14      c2 = new PdfPCell(new Phrase("Numero:"));
15      c2.setHorizontalAlignment(Element.ALIGN_CENTER);
16      c2.setColspan(2);
17      tableSMS.addCell(c2);
18      tableSMS.addCell(osms.getAddress());
19      c2 = new PdfPCell(new Phrase(osms.getMsg()));
20      ...
21      c2 = new PdfPCell(new Phrase("SMS"));
22      ...
23      c2 = new PdfPCell(new Phrase("ID"));
24      ...
25      c2 = new PdfPCell(new Phrase("Estado"));
26      ...
27      c2 = new PdfPCell(new Phrase("Time"));
28      ...
29    }
30    subCatPartSMS.add(tableSMS);
31    }
32  }
```

Listing 5.33: Parte da Classe *GeraPDF.java*

A imagem 5.5 é referente a seção do Relatório que contém os dados sobre os objetos da Classe *DadoSMS*.

1.2. SMS no Dispositivo

Numero:	962491885	SMS...SMS
SMS	sent	
ID	9339	
Estado	1	
Time	1396718320745	
Numero:	+351962491885	SMS...SMS
SMS	inbox	
ID	9338	
Estado	1	
Time	1396718238083	

Figura 5.5: Dados sobre as SMS no Relatório

5.3 Modelo de Comunicação

O fluxo de dados como se pode ver na imagem 5.6, consiste na recolha de dados no dispositivo e pela posterior transferência de dados entre as aplicações, com base no Modelo Cliente-Servidor, que culmina na criação de um relatório com os dados que foram extraídos, como se pode ver no anexo I.

Após estabelecer a ligação entre o dispositivo é o computador através do ADB, inicia-se a comunicação entra as aplicações. Para isso utiliza-se um mecanismo que possibilita a comunicação entre aplicações designado de *Socket*, com a criação de uma conexão entre elas. Neste caso utiliza-se o protocolo TCP para fazer a comunicação. Onde a aplicação Servidor define uma porta e aguarda conexões nessa porta, a aplicação Cliente solicita uma conexão e cria um canal de comunicação. O canal de ligação inicia-se quando o método *accept()* da aplicação Servidor recebe um pedido de conexão, este método retorna um *Socket* no qual é definido o *Socket* de comunicação, para encerrar a conexão utiliza-se o método *Close()*.

O processo de recolha de dados consiste na consulta dos dados por métodos definidos no SDK de desenvolvimento, *Content Provider*, o que facilita todo o processo não sendo necessário implementar código SQL (*Structured Query Language*). Sendo criados objetos referentes as Classes de dados recolhidos.

Nesta fase do processo só são criados os objetos referentes aos dados que o utilizador escolheu através da *CheckBox* na interface da aplicação.

O próximo passo passa pelo envio dos obejctos criados pelo canal. O fluxo de dados de envio e receção recorre à criação de um canal de comunicação entre as duas aplicações através da utilizado da Classe Java *ObjectOutputStream* para tratar o processo de envio e à Classe Java *ObjectInputStream* para a receção dos dados.

As classes *ObjectInputStream* e *ObjectOutputStream* permitem que objetos inteiros sejam enviados e armazenados num fluxo de *bytes*, neste caso para os objetos correspondentes entre si.

A última etapa no fluxo de dados consiste em enviar os dados recebidos para a Classe que processa os dados para gerar um relatório.

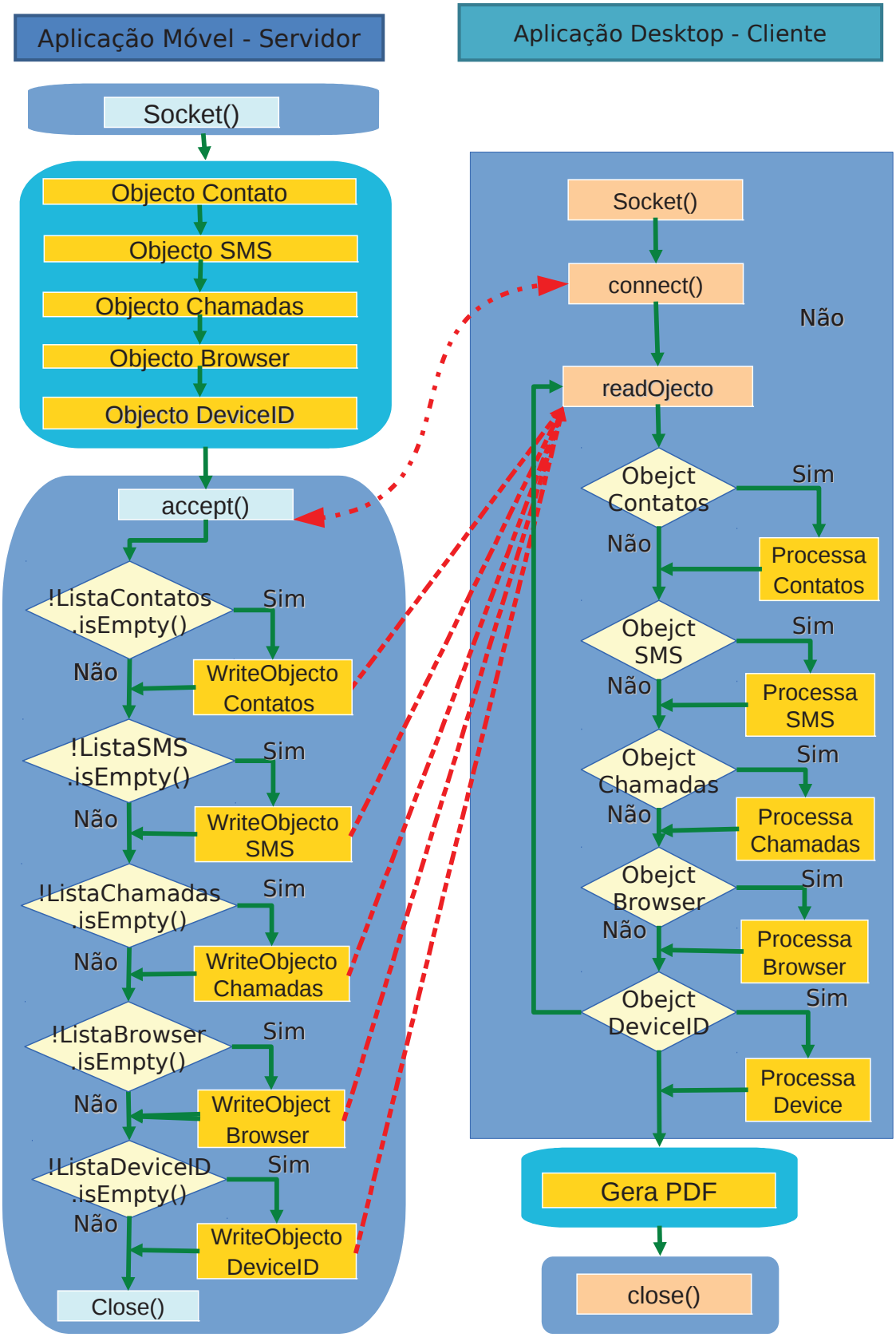


Figura 5.6: Fluxo de Dados

Capítulo 6

Testes e Análise Comparativa

6.1 Cenários

Os cenário apresentados pretendem ser o exemplo de duas situações onde se pretende fazer análise forense a dois dispositivos apreendidos. É necessário extrair dados para analisar o seu conteúdo.

Tabela 6.1: Caraterísticas dos Cenários

Cenários	**Cenário A**	**Cenário B**
Ligado	Sim	Sim
Bloqueado	Não	Sim
Acesso de Depuração	Sim	Sim
Super Utilizador	Não	Não

Será descrita a extração lógica dos dados dos dispositivos, com exemplos dos procedimentos que são necessários realizar. O objetivo é utilizar a aplicação desenvolvida em dois tipos de dispositivos, o primeiro um modelo de gama baixa, de uma empresa Chinesa e o segundo um equipamento de topo da *Samsung*. Com versões do SO diferentes e de grande utilização. Desta forma temos uma abrangência grande na utilização da aplicação. Não tendo estes cenários relação com algum caso específico real ou fictício, os dados que se pretendem extrair são os mais comuns que se podem encontrar pela utilização normal de um dispositivo móvel, como registo de chamadas, contactos, SMS, configurações do dispositivo e registos do *Browser*. Mas é importante referir que se fosse numa análise de um caso real, o investigador teria de ter informações sobre todo o processo de apreensão e ler a documentado produzida nessa etapa por forma a tomar as melhores decisões no processo de extração dos dados.

Vão ser analisados 2 dispositivos com as seguintes características.

Tabela 6.2: Caraterísticas dos dispositivos

	Cenário A	**Cenário B**
Caraterísticas	**Dispositivo 1**	**Dispositivo 2**
Fabricante	*ZTEmoblie*	*Samsung*
Modelo	A5	*Galaxy Nexus*
Versão do *Android*	2.2	4.3
Versão do *Kernel*	*2.6.32.9-perfzte-kernel@Zdroid-SMT*	*3.0.72-gfb3c9ac android-build@vpbs1. mtv.corp.google.com*
Rede	P TMN	Desconhecido
IMEI	864595000318372	351554058347080
Rede Movel	EDGE	Desconhecido
Estado	Ligado, sem bloqueio de acesso, sem permissões de super utilizador instaladas, em modo avião	Ligado, com bloqueio de acesso, sem permissões de super utilizador, não está em modo avião

6.1.1 Cenário A

No Capítulo Quinto temos a imagem 4.1 que demonstra o fluxo a ser seguido no processo de extração de dados para o cenário A e vai servir como base na descrição das etapas.

O dispositivo ao ser apreendido, independentemente da situação de apreensão, é mantido ligado e colocado em modo de avião, de forma a não estabelecer qualquer contacto com a rede. No processo inicial o investigador irá ler toda a informação referente a apreensão, neste caso os aspetos mais importantes são garantir que se trata do mesmo dispositivo que foi apreendido e verificar se foi colocado em modo de avião. Estando o dispositivo ligado e em modo de avião não é necessário isolar da rede. Não se encontrando qualquer sistema de segurança ativo que bloqueia o acesso, avançamos para o próxima etapa, neste caso não é feita nenhuma extração de dados do cartão de memoria, opção tomada porque o objetivo principal é extrair dados pela aplicação. Não se encontra nenhuma aplicação instalada que altere as permissões para super utilizador.

Assim passamos para a parte mais importante, a extração dos dados através da instalação da aplicação "*DroidExport*". Verificamos se o modo de depuração por USB se encontra ativado, através das definições do dispositivo, o que se verifica. Desta forma é possível instalar a aplicação através da ferramenta ADB disponibilizada pelo SDK do *Android*.

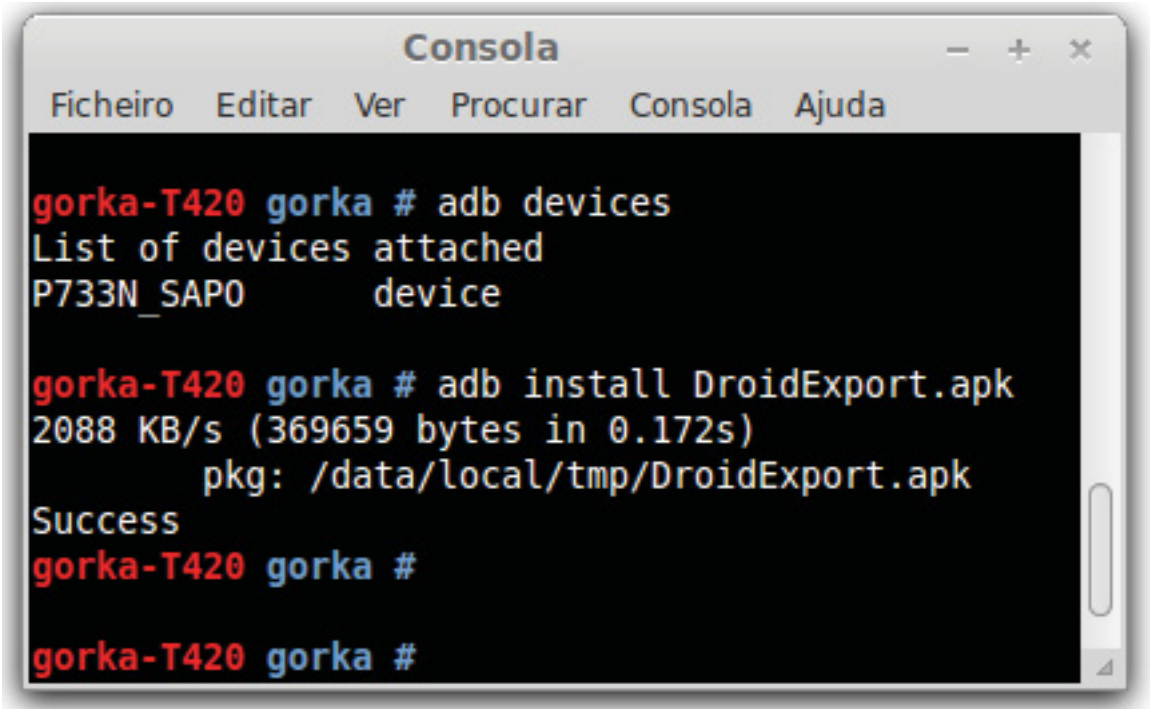


Figura 6.1: Instalação do *DroidExport* no Dispositivo 1

Com a instalação feita com sucesso 6.1, vamos executar a aplicação, como podemos ver na imagem 6.2.

Todo o processo de extração tem de ser documentado, onde são descritos todos os procedimentos tomados ao longo do processo e quais consequências dessas opções. Juntamente com os resultado obtidos, que neste caso esta no relatório produzido pela aplicação em formato PDF.

6.1.2 Cenário B

No Capítulo Quinto temos a imagem 4.2 que demonstra o fluxo a ser seguido no processo de extração de dados para o cenário B e vai servir como base na descrição das etapas.O dispositivo ao ser apreendido, independentemente da situação de apre-

Figura 6.2: Aplicação no Dispositivo 1

ensão, é mantido ligado mas não foi colocado em modo de avião, assim vai-se manter ligado a rede.

No processo inicial o investigador irá ler toda a informação referente a apreensão, neste caso um dos aspetos mais importantes é garantir que se trata do mesmo dispositivo que foi apreendido. De se seguida o investigado vai colocar o dispositivo em modo de avião, este modelo tem uma opção para se colocar em modo de avião mesmo estando bloqueado, por uma opção que existe ao pressionar o botão de Desligar. Se não fosse possível isolar da rede, a solução poderia passar por colocar o dispositivo numa sala que isole da rede até se conseguir contornar o sistema de bloqueio.

Tendo o dispositivo um sistema de segurança ativado, através de um bloqueio de acesso, vamos verificar se é possível contornar o bloqueio instalando uma aplicação via ADB. É verificado se o modo de depuração por USB se encontra ativado, através das definições do dispositivo. Desta forma é possível instalar a aplicação "*CRT-Removelocks*"e de seguida aplicação "*DroidExport*", porque já estamos ligado via ADB. A técnica utilizada para contornar o sistema de bloqueio provém de uma vulnerabilidade descoberta pela equipa de investigadores Alemães, *CureSec*, na qual é possível remover todos os bloqueios de acesso que estejam ativos no sistema. O funcionamento da técnica consiste na instalação da aplicação "*CRT-Removelocks*"e

da execução de uma comando via ADB *shell*, como se pode ver na figura6.3 [60]. Avançamos para o próxima etapa, nesta caso não é feita nenhuma extração de dados do cartão de memoria, porque o dispositivo não tem cartão de memoria e porque o objetivo principal e extrair dados pela aplicação.

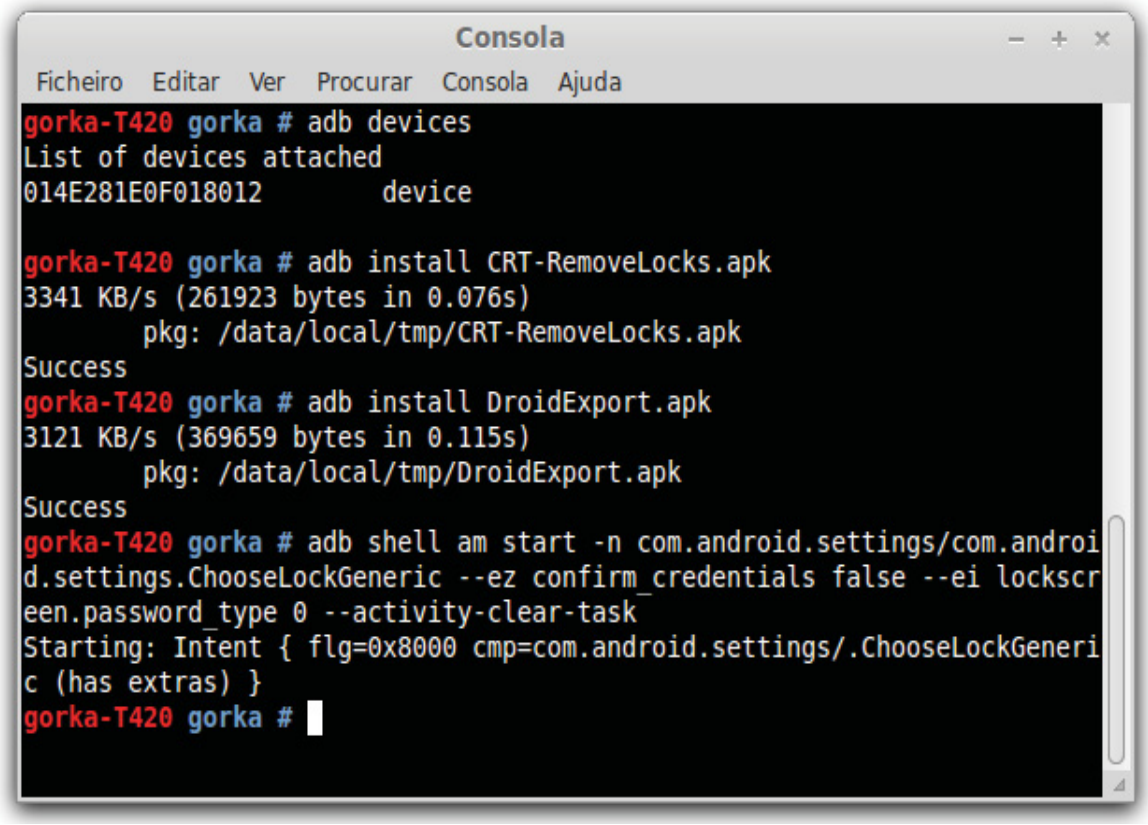


Figura 6.3: Instalação do *DroidExport* no Dispositivo 2

Com a instalação feita com sucesso 6.3, vamos executar a aplicação, como podemos ver na imagem 6.5.

Na imagem 6.4 podemos ver o bloqueio de ecrã ativo e e desativo como resultado da utilização da técnica que contorna o sistema de bloqueio.

Todo o processo de extração tem de ser documentado, onde são descritos todas os procedimentos tomados ao longo do processo e quais as consequências dessas opções. Juntamente com os resultados obtidos, que neste caso está no relatório produzido pela aplicação em formato PDF.

Figura 6.4: Sistema de Bloqueio de ecrã

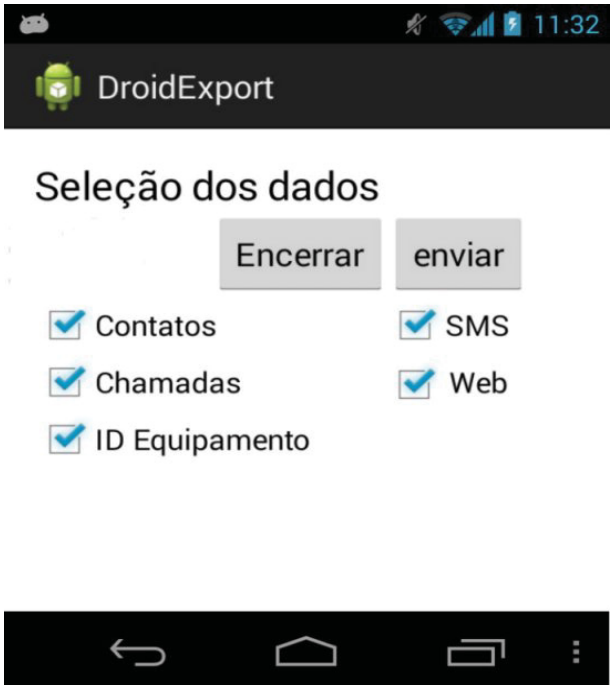


Figura 6.5: Aplicação no Dispositivo 2

6.2 Testes

Os testes são realizados seguindo as metodologia definidas anteriormente e com o objetivo de ter um parâmetro de comparação entre a aplicação desenvolvida, *DroidExport* e outra aplicação, neste caso a *viaForensica*, sendo escolhida por ser ser uma versão de utilização gratuita, de fácil instalação, reconhecida nos meios forenses e onde os resultados obtidos se podem comprar com os da aplicação desenvolvida.
Para testes temos 2 dispositivos com SO *Android*6.6, *Samsung Galaxy Nexus* com a versão ZTE A5(Dispositivo 1) e o 4.3 do *Android*(Dispositivo 2) com a versão 2.2 do *Android*. As evidências forenses que estão nos dispositivos foram inicialmente quantificados e comparadas com base nas evidencias que se poderia encontrar. É de referir que nos dispositivos existem mais evidências, mas que não foram tidas em conta para análise por não terem sido implementados métodos de extração nas aplicações em teste.
Para se realizar os testes foi utilizado uma estação de trabalho com o SO: *Linux Mint 15: Oliva x86*, equipado com um Processador: *Intel Core i5-2520M 2.50Ghz* x 2 e com 7.7 GB de memoria. Não sendo necessário uma estação de trabalho com grandes recursos.

Figura 6.6: Imagem dos Dispositivos 1 e 2 para testes

Na tabela 6.3 podemos ver as evidências que estão nos dispositivos, a contagem dos dados foi feita de forma manual, o dispositivo 1 tem um utilização real, todas os dados contidos são de uma utilização normal, não tendo sido possível contabilizar as MMS. Já o dispositivo 2 tem pouca informação, sendo um dispositivo que serviu de teste ao longo do desenvolvimento.
Ambos os dispositivos estão ligado a um operador de telecomunicações para simular toda a troca de dados, mas durante a extração estiveram em modo de voo para não alterar a informação entre a utilização das aplicações de extração. A opção de utilizar um dispositivos com informação real tem como objetivo testar as reais capacidades das aplicação desenvolvida.

Tabela 6.3: Informação nos Dispositivos 1 e 2

	Cenário A	**Cenário B**
Tipo de Dado	**Dispositivo 1**	**Dispositivo 2**
Número de Contactos	371	3
Número de SMS	986	7
Número de MMS	-	0
Número de Chamadas	500	8
Browser	32	29
Definições	Sim	Sim

Disposito 1

Tabela 6.4: Comparação entre aplicações - Cenário A

Tipo de Dado	**DroidExport**	**AFLogical**
Número de Contactos	409	90
Número de SMS	986	986
Número de MMS	0	45
Número de Chamadas	500	500
Browser	32	0
Definições	Sim	Sim

No dispositivo 1 foram instaladas ambas as aplicações, através a ferramenta ADB com o modo de depuração ativado. Para extrair os dados através da aplicação *AFLogical* é necessário clicar no ícone, selecionar todos as opções disponíveis e ativar a captura dos dados. *AFLogical* vai criar uma pasta chamada "*forensics*" onde são guardados os ficheiros extraídos, em formato CSV e num ficheiro *info.xml*. Para extrair os ficheiros que contem os dados é feita uma ligação ao dispositivo e são copiados os dados para posterior análise.

Para extrair os dados pela aplicação *DroidExport* o processo é semelhante, é necessário clicar no ícone e selecionar todas as opções disponíveis é enviar os dados para a aplicação *DroidImport*, que ao receber vai criar um ficheiro PDF que contem o relatório com uma listagem dos dados. É possível verificar o número de dados extraídos pela aplicação DroidImport pela linha de comandos, como se pode ver na imagem 6.7.

Comparando os dados extraídos, pelas duas aplicações, podemos chegar a seguinte conclusão. Ambas completam o processo extraindo as informações necessárias, sendo que a grande diferença se encontra nos dados dos Contactos, no *DroidExport* temos 409 contatos e no *AFLogical* 90 contatos. Verificando os dados da tabela 6.3, também encontramos uma diferença relativa ao número de contactos registados, sendo 371. Esta divergência levanta uma questão, o porque de serem divergente. A justificação é simples, a aplicação *DroidExport* tem a capacidades de extrair os contactos das contas de *email* que estão configurada no dispositivo, ao contrario da *AFLogical*, que só consegui extrair 90, valor muito baixo tendo em conta o número de contactos. É no dispositivo não conseguimos visualizar os contactos das contas de *email*, só contatos registados pelo gestor de contatos. Quando aos dados do *Browser* só a *DroidExport* é que estava preparada para extrair o histórico de utilização e os *bookmarks*. Nas MMS só a aplicação *AFLogical* esta preparada para extrair.

É salientar que o processo de extração no disposito 1 é mais lento devido a ter menos capacidade de processamento e ter muita informação.

Disposito 2

Tabela 6.5: Comparação entre aplicações - Cenário B

Tipo de Dado	**DroidExport**	**AFLogical**
Número de Contactos	8	3
Número de SMS	7	7
Número de MMS	0	45
Número de Chamadas	8	8
Browser	29	0
Definições	Sim	Sim

No dispositivo 2 foram instaladas ambas as aplicações, através a ferramenta ADB com o modo de depuração ativado. Para extrair os dados pela aplicação *AFLogical* é necessário clicar no ícone, selecionar todos as opções disponíveis e ativar a captura dos dados. *AFLogical* vai criar uma pasta chamada "*forensics*" onde são guardados

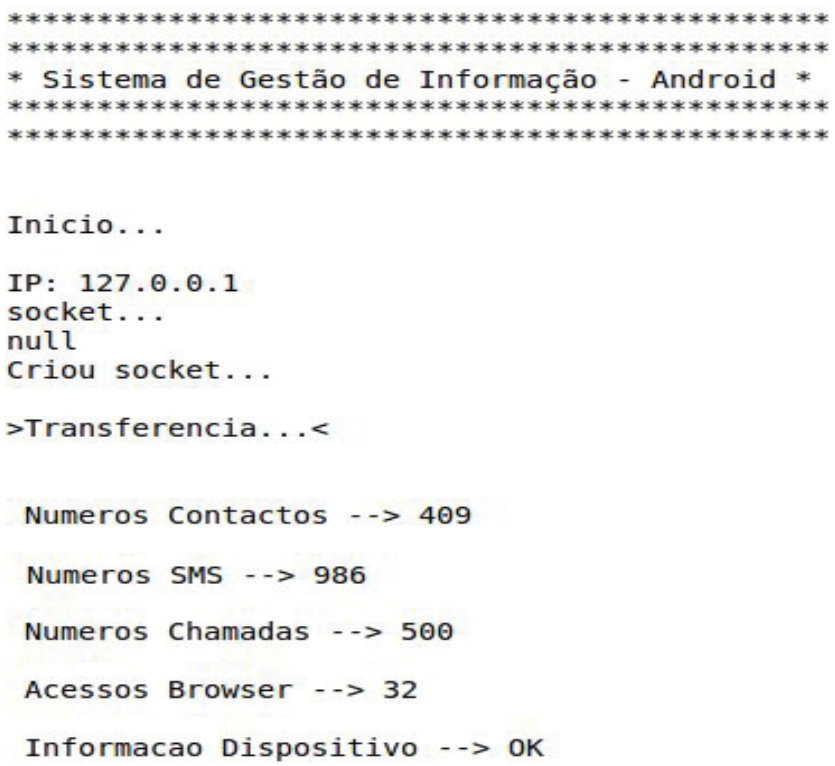


Figura 6.7: Dados extraídos Dispositivos 1

os ficheiros extraídos, em formato CSV e num ficheiro info.xml. Para extrair os ficheiros que contem os dados é feita uma ligação ao dispositivo e são copiados os dados para posterior análise.

Para extrair os dados pela aplicação *DroidExport* o processo é semelhante, é necessário clicar no ícone e selecionar todas as opções disponíveis é enviar os dados para a aplicação *DroidImport*, que ao receber vai criar um ficheiro PDF que contem o relatório com uma listagem dos dados. É possível verificar o número de dados extraídos pela aplicação *DroidImport* pela linha de comandos, como se pode ver na imagem 6.8.

Comparando os dados extraídos, pelas duas aplicações, podemos chegar a mesma conclusão que na obtivemos com a Aplicação 1. Com exceção dos dados obtidos nas MMS pela aplicação *AFLogical*, visto não ter encontrado informação sobre este tipo de dados.

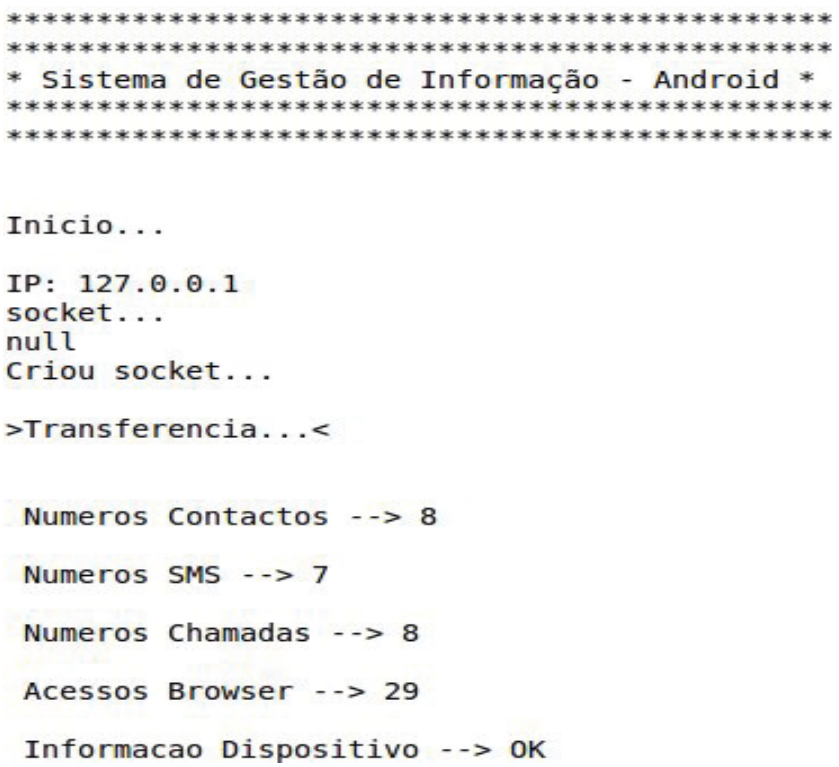


Figura 6.8: Dados extraídos Dispositivos 2

6.3 Comparação com outras soluções

Das poucas soluções que existem no momento para realizar uma extração de dados de um dispositivo com SO *Android* saliento cinco soluções, duas de utilização livre e três proprietárias.

***Cellebrite* UFED** A *Cellebrite* UFED é uma solução forense para dispositivos *Android*, sendo uma aplicação proprietária, mas que apresenta funcionalidades exclusivas.

- Tem um método de extração física para mais de 200 dispositivos baseados em *Android*, sendo capaz de ultrapassar sistema de bloqueio (padrão/PIN (*Password Identification Number*)/*password*) e utiliza sistemas de *boot* patenteados pela *Cellebrite*, o que permite um processo de extração único e adequado para investigação forense. A extração física pode ser feita independentemente da versão do SO;

- Extração física e descodificação e feita através de depuração via USB para todas as versões de *Android*, incluindo o *Android* 4.X . A extração física de qualquer dispositivo só é possível se o modo de depuração por USB estiver activado.

Suporte a dados de aplicações: A extração lógica de dados para as aplicações instaladas nos dispositivos *Android* como *Facebook, Google+, Skype, Twitter, Viber, Yahoo Messenger, Whatsapp*, entre outros. O sistema UFED consiste num dispositivo portátil, que contem um conjunto de *software* de extração e análise, cabos de dados, adaptadores e outros periféricos. Não sendo necessário qualquer *software* ligado a um PC para executar a extração e análise. O UFED tem integrado no hardware um leitor de cartão SIM, juntamente com opções de ligações sem fios.

O sistema UFED é disponibilizado apenas para entidades governamentais e organizações empresariais. UFED extrai dados de dispositivos móveis diretamente para um cartão SD ou *flash drives* USB. Uma das grandes capacidade do UFED e de conseguir quebrar códigos, decifrar informações cifradas, e adquirir dados ocultos e excluídos.

O UFED recebeu a nomeação de "*Phone Forensic Hardware Tool of the Year*"por quatro anos consecutivos pelo "*Forensic 4cast Awards*" [61]. O UFED é apresentado como tendo a maior cobertura para disponível moveis, com a capacidade para extrair dados de quase 8.200 dispositivos. Estão incluídos *smartphones*, PDAs, telemóveis e *tablet*. O UFED pode extrair, descodificar, analisar contactos da agenda, todos os tipos de conteúdo multimédia, mensagens SMS e MMS, *logs* de chamadas, ESN, IMEI e SIM informações de localização de ambos não-volátil memória e armazenamento volátil igualmente, em vários idiomas internacionais. Suporta todos os protocolos, incluindo telemóveis CDMA , GSM, IDEN (*Integrated Digital Enhanced Network*), e TDMA (*Time Division Multiple Access*), e também pode interagir com os sistemas de ficheiros de diferentes SO, como iOS , *Android, BlackBerry, Symbian, Windows Mobile e Palm*, bem como SO de telemóveis.

A Extração física permite recuperar informações excluídas, decifrar dados cifrados, adquirir informações protegidas por *password*, aplicações móveis.

A série UFED vem em três versões diferentes:

- UFED *Ultimate* – contem as funcionalidades de todas as outras versões UFED, permitindo a extração física, lógica, a *password* do utilizador, extração do sistema de ficheiros, a extração de dados, ocultos e excluídos, decifrar códigos de bloqueio e acesso e descodificar dados de aplicações internos. IMSI registo,

cartões SIM, histórico de utilização. UFED *Chinex*, permite a extração física e descodificação para diapositivos chineses.

- UFED *Physical Analyzer* é um pacote de *software* que é utilizado para descodificar e analisar imagens físicas de dispositivos móveis .

- UFED *Logical* permite a extração de dados lógico de smartphones e telemóveis mais antigos.

Cellebrite é muitas vezes a ferramenta utilizada pelas forças policias nas suas investigações. O NIST testou por duas vezes *Cellebrite* UFED como parte do seu projecto de "*Computer Forensics Tool Testing Project*", em 2009 [62] e 2010 [63]. Em ambos os testes os resultados obtidos demonstram que na globalidade o *software* se encontra dentro do padrões da NIST para análises a dispositivos moveis [64] [65].

***Secure View* 3** A *Secure View* 3 é uma solução proprietária da *Susteen, Inc.* que apresenta a vantagem de ter um custo de aquisição reduzido. O *Secure View* 3 permite extrair um conjunto de dados para o investigador forense através de uma metodologia forense que garante que todos os procedimentos são executados de forma correta, sem alteração dos dados. O processo tem por base três procedimentos, Aquisição, Análise e Relatório. Existem 4 módulos que permitem aumentar as capacidade de extração de dados e a sua capacidade de melhorar os processo de investigação.

- SvDDR - *Deleted Data Recovery* – Ferramenta de aquisição para SO *Android*: *Secure View* e o modulo SvDDR descodifica a extração de dados para os utilizadores de forma inteligente sem a necessidade de recorrer a técnicas de programação avançadas;

- SvSmart - *Intelligence Gathering Tool*: Ferramenta de extração e triagem. Permite racionalizar o tempo da investigação, utilizando condições pré-definidas para extrair e analisar ao mesmo tempo;

- svPin - Desbloqueio CDMA: Ferramenta de Aquisição de *password*: O *Secure View* adquire o código PIN por um processo simples de 3 etapas;

- SvLoader - Análise de Componentes: O *svLoader* integra documentos CVS (*Concurrent Version System*) (XRY, UFED, *Katana* entre outros), produzindo relatórios detalhados.

O *Secure View* 3 permite ter acesso a funcionalidades únicas para a investigação forense:

- *Discovery*: Pesquisa por palavras-chave em vários tipo de conteúdo nos dados extraídos e vários aparelhos ao mesmo tempo;
- *Timeline*: Análise temporal em todos os conjuntos de dados (registos de chamadas, SMS/MMS , contacto, calendário e outros dados, como histórico da *web*);
- Visualização Gráfica: Atividade do dispositivo (chamadas e textos) ou contactos (registo de contactos);
- Mapa Atividade: Ilustração gráfica das atividades do dispositivo (chamadas e textos);
- Galeria: Identificar a data, hora, nome do ficheiro, e se possível, coordenadas GPS com etiquetas geo META das fotos tiradas;
- Atividade *Web*: Registo de toda a atividade *web* por hora, a data e *links* do site, incluindo a atividade nas rede social [66] [67].

Paraben – *Device Seizure* A *Paraben Corporation* disponibiliza o *software* forense proprietário *Device Seizure*, que é um sistema de extração e análise forense para dispositivos moveis. Com recurso de análise, extração física e lógica, extração de ficheiros. Contem um sistema inteligente de análise de dados, com uma constante atualização para as novas versões dos SO. Estas funções são possíveis através de um cabo de dados USB padrão com qualquer PC.

Visão geral do *software*:

- Extrações lógica e física, análise avançada, integração de *Google Earth*, classificação de ficheiros e relatórios abrangentes;
- Extração de dados lógico: Os dados do utilizador, tais como registos de chamadas, SMS , contactos, fotos , entre outros, podem ser facilmente adquiridos através apreensão do dispositivo ou DDS (*Deployable Device Seizure*);
- Extração de dados Físicos: Extração física completo, incluindo sistema de ficheiros e dados apagados, incluindo a maioria dos telemóveis CDMA, SO *Android*, e alguns dispositivos GPS;

- Extração das *password* do utilizador: Recuperação da *password* do utilizador do dispositivos móveis ou ultrapassar sistema de bloqueio. Apreensão dispositivo extrai senhas de usuários de centenas de dispositivos;

- Integração com o *Google Earth*: Dados de GPS podem ser extraídos a partir de dispositivos de GPS, bem como de *smartphone*. Na apreensão do dispositivo é possível visualizar estas coordenadas GPS facilmente por meio da integração com o *Google Earth*;

- Análise de Dados: Extrair dados de dispositivos moveis é apenas a primeira fase. È preciso ser capaz de analisar esses dados. É preciso ter forma exibir os dados do utilizador e recuperar dados apagados de ambos os ficheiro lógicos e extrações físicas.

A lista completa dados que podem ser adquiridos a partir de modelos de *Android*:

- Endereços incluindo contactos grupos, organizações e endereços Configurações, nome, número de telefone e endereço

- Mensagens SMS, MMS

- Histórico de chamadas

- Métodos de contacto

- Histórico do *browser*

- Imagem(metadados)

- Imagem *Thumbnail* (metadados)

- Áudio

- Vídeos (metadados)

- Lista de todos as aplicações instalados e sua a versão [68];

ADEL (*Android* Data Extractor Lite) ADEL: *Android Data Extractor Lite* é uma ferramenta gratuita de extração de dados desenvolvida para funcionar nas versões 2.x, sendo capaz de extrair os ficheiro da base de dados SQLite e o conteúdo armazenado nos ficheiros de *dumping*. O desenvolvimento da aplicação teve por base as seguintes diretrizes:

- Princípios Forense: tratar os dados de forma forense correta. Este objetivo é alcançado pelo fato de que as atividades não são realizadas diretamente no *smartphone*, mas em uma cópia da base de dados. Este procedimento garante que os dados não são alterados. Para provar a veracidade forense, os valores de *hash* são calculados antes e depois de cada análise;

- Escalabilidade: foi construído de forma modular e contém dois módulos separados: modulo de análise e o módulo de relatório. Existem interfaces predefinidas entre esses módulos e ambos podem ser facilmente alterada por funções adicionais. A estrutura modular permite a análise de dumping e outras bases de dados de *smartphones*, e facilita as atualizações do sistema no futuro;

- Usabilidade: O uso da ADEL é destinado a ser o mais simples possível para permitir a sua utilização tanto por pessoas qualificadas e não-especialistas. Na melhor das hipóteses, a análise do smartphone é feita de uma forma automática. Além disso, o módulo de relatório cria um relatório pormenorizado, incluindo todos os dados descodificados. Durante a execução é possível gerar um arquivo de *log*, onde todos os passos importantes são registados. ADEL utiliza o *Kit* SDK e o ADB para enviar os ficheiros da base de dados.

Dados extraídos :

1. Informações do dispositivo e do cartão SIM (IMEI,IMSI,número de série, etc);
2. Lista de contactos e de chamadas;
3. Dado do calendário;
4. Mensagens SMS;
5. Localizações GPS.

Os dado obtidos são gravados num ficheiro XML e processados pelo módulo de relatório para facilitar a criação do relatório, sendo criado um ficheiro XML para cada tipo de dado [69].

***AFLogical-OSE* (Maquina virtual *Santoku*)** *AFLogical*^{TM} é uma ferramenta gratuita de análise forense desenvolvido pela *viaForensics*. *AFLogical*^{TM} realiza uma aquisição lógica de qualquer dispositivo com uma versão do SO *Android* 1.5 ou superior. Os dados extraídos são gravados no cartão SD do examinador no formato CSV que facilmente se importa, facilitando o processo de extrair e analisar os dados do dispositivo.

A *viaForensics* disponibiliza três tipo de aplicações:

- *viaExtract* - está disponível como parte do software proprietária da *viaForensics*. permite uma análise mais profunda aos dispositivos *Android*, comparando com qualquer outro produto no mercado. Produzindo um relatório extremamente detalhado.

Tabela 6.6: Informação extraida pela Aplicação *viaExtract*

Browser Bookmarks	*Browser Searches*	*Calendar Attendees*
Calendar Events	*Calendar Extended Properties*	*Calendar Reminders*
Calendars	*CallLog Calls*	*Contacts*
Contacts Extensions	*Contacts Groups*	*Contacts Organizations*
Contacts Phones	*Contacts Settings*	*External Image Media*
External Image Thumb Media	*External Media*	*External Videos*
IM Account	*IM Accounts*	*IM Chats*
FIMContactsProvider	*IM Invitations*	*IM Messages*
Internal Image Thumb Media	*Internal Videos*	*Maps-Friends*
Maps-Friends contacts	*Maps-Friends extra*	MMS
MmsPartsProvider	*Notes*	*People*
People Deleted	*PhoneStorage*	*Search History*
SMS	*Social Contracts Activities*	

- *AFLogical OSE - AFLogical* ™ *Open Source Edition* é uma versão gratuita do *software* disponível através do *Google Code*. Permite extrair todas as MMS disponíveis, SMS, contatos e registros de chamadas a partir de seu dispositivo *Android*. *AFLogical* ™ OSE também está disponivel no *Santoku-Linux*, que é um SO dedicada à análise forense para dispositivos móveis, análise de *malware* e testes de segurança.

- *AFLogical LE* - A *AFLogical Law* enforcement permite extrair todos os dados lógicos de um dispositivo *Android* mas não contém a interface *viaExtract* GUI.

Uma das características interessante da *ViaForensics* é ser de utilização livre por agências policiais. *ViaForensics* reconheceu a necessidade dar formação para isso disponibiliza sessões de formação para a utilização das suas ferramentas, juntamente com o resultado das suas pesquisas na área forense A solução *AFLogical*

permite extrair os dado mais gerais o que oferece o melhor retorno numa relação de investimento, tempo e resultados [70] [71].

Capítulo 7

Conclusão

7.1 Conclusão

Os dispositivos móveis entram nas nossas vidas permitindo uma evolução na nossa forma de nos comunicarmos com o mundo, a união entre o típico telemóvel e um computador deu origem aos *smartphone* e ao *tablet*, dispositivos com os quais facilmente nos ligamos a web, enviamos SMS, acedemos a aplicações, entre outras funcionalidades. Sendo o *Android* o SO Mais utilizado no dispositivos, isto por dois fatores chave, ser um SO *OpenSource* e a capacidade de ser instalado em diversos tipo de *hardware*.

Os dispositivos móveis consegue gerar e armazenar um conjunto de informações que são importantes quando se realiza uma investigação forense. Mas para isso é importante que o investigador tenha conhecimento da melhor forma de realizar a a investigação, tendo em conta as circunstâncias em que o dispositivo se encontra.

Para isso tem de seguir uma metodologia que permita obter as evidencias que ajudem na investigação. Para analisar o Sistema Android abrangendo todas as circunstâncias que se podem encontrar num dispositivos, após uma investigação as metodologias e técnicas opto-se por utilizar a metodologia de André Simão, porque consegue abranger todas as circunstâncias que existe até ao momento, indicando o caminho a seguir até se conseguir extrair as dados.

Após uma análise ao estado da arte, pode-se concluir que a maioria dos estudos que tem sido realizados se focam em processos de extração Física, obtendo na globalidade todo o sistema, o que obriga a investigar um conjunto grande de dados. O processo desenvolvido consiste numa extração Lógica, que permite ao investigador escolher os dados a extrair, o que acelera e facilita o processo, porque se restringe a procura a uma conjunto de dados.

Durante a implementação da Aplicação Servidor e Cliente existiram uma série de dificuldades, maioritariamente relacionadas com a necessidade do funcionarem em todas as versões do SO *Android* e a configuração do sistema para o envio e receção dos dados, através de uma programação orientada a objetos, mas que resulta numa aplicação mais robusta, funcional e com uma grande capacidade de evolução. Sendo essa capacidade de evolução uma mais valia da aplicação, isso porque o Sistema *Android* tem apresentado uma grande evolução tecnologia, obrigando a uma adaptação da aplicação as novas e as já existes funcionalidades. É importante ter em atenção um pormenor em relação as novas API, porque alguns dos novos métodos vão deixar outros obsoletos, o que por vezes obriga a uma decisão de programação para se consiga abranger todos os sistema e que por vezes se opte por extrair só os dados que sejam comuns a todos os SO Android, ou então optar por implementar métodos específicos.

Nos testes realizados foi possível comprovar que funciona e consegue extrair os dados que foram selecionados em ambos dos dispositivos. Existindo por vezes problemas de comunicação entre os dois dispositivos, por não se executar o procedimento de extração da forma correta, primeiro executar a aplicação Servidor e depois a aplicação Cliente, o que origina erros de comunicação.

Sobre o relatório produzido, podemos concluir que resulta numa mais valia comparando com a solução da AFLogical, porque não é necessário estar a importar os dados para serem formatos e apresentados de forma legível, são automaticamente exportados para um documento em PDF formatados em tabelas que permitem ler. vantagens no caso da extração dos contactos e na criação do relatorio.

7.2 Trabalho Futuro

A Computação Forense em SO Android ainda tem muito para evoluir, principalmente no que respeita a ferramentas *Open Source*, quando a metodologia utilizada será sempre um ponto de partida, para futuras pesquisas. No que respeita ao estado da arte, será importante investigar os avanços na independência do tipo de plataforma e nas soluções para analisar os dados extraídos, que estão muito pouco desenvolvidas. Outro ponto importante são as técnicas anti-forense, é importante fazer uma investigação das técnicas que existam para SO *Linux* ou outros SO e que podem ser adaptadas para o SO *Android*.

Como trabalho futuro, a nível do desenvolvimento da aplicação, propõe-se uma evolução da aplicação no que respeita aos dados a extrair. Procurando abranger

um conjunto maior de dados, indo ao encontro do que as sistemas proprietários conseguem extrair. Atualizar o sistema para as novas versões que vão surgir, tendo atenção aos métodos que podem ficar obsoletos. Na parte da aplicação Cliente, o que se pretende é ter uma aplicação mais evoluída que consiga trabalhar os dados recebidos de uma forma mais interativa. Com isto pretende-se que a aplicação tenha novas funcionalidades para a visualização dos dados, como por exemplo a criação de *time lines* para cada contacto, pesquisa por informação, análise de dados de "*apps*", exportação para outros formatos, desbloqueio de sistemas de segurança, relatórios personalizados, entre outras funcionalidades. No geral que se caminhe para que esta solução consiga ser uma alternativa as soluções proprietárias.

Bibliografia

[1] Infopédia. Porto Editora. [*Online*]. Disponível: http://www.infopedia.pt/pesquisa-global/forense (citado na pág. 5)

[2] US-CERT, "Computer forensics us-cert," 2008. (citado na pág. 5)

[3] A. Toffler, "A road map for digital forensic research," *DFRWS Techinical Report*, 2001. [*Online*]. Disponível: http://www.dfrws.org/2001/dfrws-rm-final.pdf (citado na pág. 5)

[4] D. Farmer e W. Venema. (2006, 07) Forensic discovery—chapter 1: The spirit of forensic discovery. [*Online*]. Disponível: http://www.porcupine.org/forensics/forensic-discovery/chapter1.html (citado na pág. 6)

[5] B. D. Carrier. (2006, 07) Basic digital forensic investigation concepts. [*Online*]. Disponível: http://www.digital-evidence.org/di_basics.html (citado na pág. 6)

[6] J. Haggert. Computer forensics process. [*Online*]. Disponível: http://www.cms.livjm.ac.uk/cmpjhagg/forprocess.htm (citado na pág. 8)

[7] SecureState. (2014) Our approach and methodology. SecureState. [*Online*]. Disponível: http://www.securestate.com/Services/Incident20Response/Pages/Forensic-Analysis.aspx (citado na pág. 8)

[8] R. Rivest. (1992) The md5 message-digest algorithm. MIT Laboratory for Computer Science. [*Online*]. Disponível: http://tools.ietf.org/html/rfc1321 (citado na pág. 8)

[9] M. T. Jones. (2007) Anatomia do sistema de arquivos do linux. [*Online*]. Disponível: https://www.ibm.com/developerworks/br/library/l-linux-filesystem/ (citado na pág. 9)

[10] J. D. Mancilha, "Análise forense em ambiente linux e windows: Uma atualização teórica," 2011. [*Online*]. Dis-

ponível: http://fatecsjc.edu.br/trabalhos-de-graduacao/wp-content/uploads/2012/04/Analise-Forense-em-Ambiente-Linux-e-Windows.pdf (citado na pág. 9)

[11] NETMarketShare. (2013) Desktop operating system market share. [*Online*]. Disponível: http://www.netmarketshare.com/operating-system-market-share.aspx?qprid=8&qptimeframe=M&qpsp=176&qpcustomd=0&qpcd=1300 (citado na pág. 11)

[12] J. R. Kaveesh Dashora, Deepak Singh Tomar, "A practical approach for evidence gathering in windows environment," *International Journal of Computer Applications*, vol. Volueme 5, 2010. (citado na pág. 12)

[13] V. Derrick J. Farmer Burlington, "A windows registry quick reference: For the everyday examiner." [*Online*]. Disponível: http://www.forensicfocus.com/downloads/windows-registry-quick-reference.pdf (citado na pág. 13)

[14] ACPO, *Good Practice Guide for Computer-Based Electronic Evidence Official release version*, official release version ed., Association of Chief Police Officers (ACPO), www.acpo.police.uk. [*Online*]. Disponível: http://www.7safe.com/electronic_evidence/ACPO_guidelines_computer_evidence.pdf (citado nas págs. 14, 23, 25 e 38)

[15] B. Pladna, "Computer forensics procedures, tools, and digital evidence bags: What they are and who should use them," 2008. [*Online*]. Disponível: http://www.infosecwriters.com/text_resources/pdf/BPladna_Computer_Forensic_Procedures.pdf (citado na pág. 14)

[16] S.-J. Wang, "Measures of retaining digital evidence to prosecute computer-based cyber-crimes," *ACM DL*, vol. Volume 29 Issue 2,, pp. Pages 216–223, 2007. [*Online*]. Disponível: http://dl.acm.org/citation.cfm?id=1222980 (citado na pág. 15)

[17] P. Mell e T. Grance, *The NIST Definition of Cloud Computing*, National Institute of Standards and Technology (NIST), 2011. [*Online*]. Disponível: http://csrc.nist.gov/publications/nistpubs/800-145/SP800-145.pdf (citado na pág. 16)

[18] K. Scarfone, *Guidelines for Managing the Security of Mobile Devices in the Enterprise*, nist special publication ed., NIST, 2013. [*Online*]. Disponível: http://nvlpubs.nist.gov/nistpubs/SpecialPublications/NIST.SP.800-124r1.pdf (citado na pág. 19)

[19] A. Spadari. Sistemas operacionais para celulares e dispositivos móveis. [*Online*]. Disponível: http://pt.kioskea.net/faq/11106-sistemas-operacionais-para-celulares-e-dispositivos-moveis (citado na pág. 20)

[20] W. Jansen e R. Ayers, *Guidelines on Cell Phone Forensics*, NIST - National Institute of Standards and Technology Std., 2007. (citado na pág. 22)

[21] E. Casey, *Digital Evidence and Computer Crime - Forensic Science, Computers and the Internet*, A. Press, Ed. 3rd Edition, 2011. (citado na pág. 23)

[22] U. D. of Justice Office of Justice Programs, *Electronic Crime Scene Investigation: A Guide for First Responders*, second edition ed., National Institute of Justice, 2008. (citado nas págs. 25, 37, 38 e 44)

[23] *Searching and Seizing Computers and Obtaining Electronic Evidence in Criminal Investigations*, United States Department of Justice, 2002. [*Online*]. Disponível: http://www.mekabay.com/methodology/sscoevci_doj_2002.pdf (citado na pág. 25)

[24] G. I. Android. (2012) the world's most popular mobile platform,. [*Online*]. Disponível: http://developer.android.com/about/index.html (citado nas págs. 29, 31 e 32)

[25] Licenses. [*Online*]. Disponível: http://source.android.com/source/licenses.html (citado na pág. 29)

[26] D. Ehringer, "The dalvik virtual machine architecture," 2010. (citado na pág. 31)

[27] Plataform versions. [*Online*]. Disponível: http://developer.android.com/about/dashboards/index.html#Platform (citado na pág. 32)

[28] uses-sdk. [*Online*]. Disponível: http://developer.android.com/guide/topics/manifest/uses-sdk-element.html (citado na pág. 34)

[29] Yaffs licence faqs. [*Online*]. Disponível: http://www.yaffs.net/yaffs-licence-faqs (citado na pág. 34)

[30] Supported media formats. [*Online*]. Disponível: http://developer.android.com/guide/appendix/media-formats.html (citado na pág. 34)

[31] U. Europeia, Ed., *Diretiva 2006/24/CE do Parlamento Europeu e do Conselho*, União Europeia, 2006. [*Online*]. Disponível: http://eur-lex.europa.eu/legal-content/PT/TXT/?uri=CELEX:32006L0024 (citado nas págs. 35 e 36)

[32] T. de Justiça da União Europeia, *Acórdão do Tribunal de Justiça(Grande Secção) 8 de abril de 2014*, Grande Secção Std., Abril 2014. [*Online*]. Disponível: http://curia.europa.eu/juris/document/document.jsf;jsessionid=9ea7d2dc30d511e1a385f52844d591cf8eaa020b3ebe.e34KaxiLc3qMb40Rch0SaxuNbNb0?text=&docid=150642&pageIndex=0&doclang=pt&mode=req&dir=&occ=first&part=1&cid=144701 (citado na pág. 36)

[33] D. S. Ramalho e J. D. Coimbra, "A declaração de invalidade da diretiva 2006/24/ce," 2014. [*Online*]. Disponível: http://www.servulo.com/xms/files/publicacoes/Updates_2014/Update_TI_DSR_JDC_A_declaracao_de_invalidade_da_diretiva_2006_24_CE_10_04_2014.pdf (citado na pág. 36)

[34] M. D. Masseno, *Que fazer, na UE, depois do Acórdão Digital Rights Ireland?*, Laboratório UbiNET Std., 5 2014. (citado na pág. 36)

[35] W. Jansen e R. Ayers, *Guidelines on Cell Phone Forensics*, National Institute of Standards and Technology, NIST, 2007, recommendations of the National Institute of Standards and Technology. (citado nas págs. 36, 37, 38 e 44)

[36] A. Hoog. (2010) Open source android digital forensics application. [*Online*]. Disponível: http://computer-forensics.sans.org/blog/2010/03/01/ (citado nas págs. 41 e 45)

[37] M. Spreitzenbarth., "Panoptes: An android forensic software agent. technical report," *University of Mannheim*, 2010. (citado nas págs. 41 e 45)

[38] Y. Lai, C. Yang, C. Lin, e T. Ahn, "Design and implementation of mobile forensic tool for android smart phone through cloud computing," *ICHIT 2011*, vol. CCIS 206, p. 196–203, 2011. (citado nas págs. 47 e 52)

[39] Android. [*Online*]. Disponível: http://developer.android.com/index.html (citado na pág. 47)

[40] A. Hoog, *Android Forensics - Investigation, Analysis and Mobile Security for Google Android*, J. McCash, Ed. SYNGRESS, 2011. (citado na pág. 48)

[41] T. Vidas, C. Zhang, e N. Christin, "Towards a general collection methodology for android devices," 2011. (citado nas págs. 49 e 50)

[42] D. Votipka, T. Vidas, e N. Christin, "Passe-partout: a general collection methodology for android devices," *IEEE*, 2013. (citado na pág. 49)

[43] J. Grover, "Android forensics: Automated data collection and reporting from a mobile device," *Digital Investigation*, vol. S12–S20, 2013. (citado nas págs. 50 e 51)

[44] N. Son, Y. Lee, D. Kim, J. James, S. Lee, e K. Lee., "A study of user data integrity during acquisition of android devices," *Digital Investigation*, vol. S3-S11, 2013. [*Online*]. Disponível: http://www.sciencedirect.com/science/journal/17422876 (citado nas págs. 50 e 52)

[45] J. Stüttgen e M. Cohen, "Anti-forensic resilient memory acquisition," *Digital Investigation*, vol. S105–S115. [*Online*]. Disponível: http://www.sciencedirect.com/science/journal/17422876 (citado na pág. 50)

[46] G. Grispos, W. B. Glisson, e T. Storer, "Using smartphones as a proxy for forensic evidence contained in cloud storage services," *46th Hawaii International Conference on System Sciences*, 2013. (citado na pág. 51)

[47] C. Racioppo e N. Murthy, "Android forensics: A case study of the "htc incredible," *Proceedings of Student-Faculty Research Day, CSIS*, 2012. (citado nas págs. 51 e 53)

[48] A. Goel, A. Tyagi, e A. Agarwal, "Smartphone forensic investigation process model," *International Journal of Computer Science & Security (IJCSS)*, vol. 6, 2012. (citado na pág. 51)

[49] A. Simão, F. Sícoli, L. Melo, F. Deus, e R. Júnior, "Acquisition and analysis of digital evidence in android smartphones," *The International Journal of FORENSIC COMPUTER SCIENCE, IJoFCS*, vol. 1, pp. 28–43, 2011. [*Online*]. Disponível: http://dx.doi.org/10.5769/J201101002 (citado na pág. 52)

[50] N. C. Timothy Vidas, Chengye Zhang, "Towards a general collection methodology for android device," 2011. (citado na pág. 52)

[51] A. Aviv, K. Gibson, E. Mossop, M. Blaze, e J. Smith, "Smudge attacks on smartphone touch screens," 2010. (citado na pág. 52)

[52] M. Yates, "Practical investigations of digital forensics tools for mobile devices," 2011. (citado na pág. 53)

[53] S. Azadegan, W. Yu, H. Liu, M. Sistani, e S. Acharya, "Novel anti-forensics approaches for smart phones," *45th Hawaii International Conference on System Sciences, IEEE*, vol. 2012, 2012. (citado na pág. 53)

[54] X. Lee, C. Yang, S. Chen, e J. Wu, "Design and implementation of forensic system in android smart phones," *Taiwan: Networks and Multimedia Institute for Information Industry*, 2010. [*Online*]. Disponível: http://security.nknu.edu.tw/publications/2010JWIS_Android.pdf (citado na pág. 53)

[55] A. Simão, *Proposta de Método para Análise Pericial em Smartphone com SO Android*, Universidade de Brasília, 2011. (citado na pág. 54)

[56] Get the android sdk. [*Online*]. Disponível: http://developer.android.com/sdk/index.html (citado na pág. 59)

[57] [*Online*]. Disponível: http://developer.android.com/tools/sdk/eclipse-adt.htm (citado na pág. 59)

[58] Andrdoid debug bridge. [*Online*]. Disponível: http://developer.android.com/tools/help/adb.html (citado na pág. 60)

[59] itextandrdoid debug bridge. [*Online*]. Disponível: http://itextpdf.com/ (citado na pág. 60)

[60] C. GmbH. (2013) Cve-2013-6271: Remove device locks from android phone. The Curesec GmbH. [*Online*]. Disponível: https://cureblog.de/2013/11/cve-2013-6271-remove-device-locks-from-android-phone/ (citado na pág. 91)

[61] L. Whitfield. (2012) Forensic 4cast awards 2012 – results. [*Online*]. Disponível: https://forensic4cast.com/2012/06/forensic-4cast-awards-2012-results/ (citado na pág. 98)

[62] U. D. of Justice, "Test results for mobile device acquisition tool:cellebrite ufed 1.1.05," 2009. [*Online*]. Disponível: https://www.ncjrs.gov/pdffiles1/nij/228220.pdf (citado na pág. 99)

[63] E. Holder, L. Robinson, e J. HaggertLaub, "Test results for mobile device acquisition tool: Cellebrite ufed 1.1.3.3 – report manager 1.6.5," 2010. [*Online*]. Disponível: https://ncjrs.gov/pdffiles1/nij/231987.pdf (citado na pág. 99)

[64] Análise forense de android. [*Online*]. Disponível: http://www.cellebrite.com/pt/mobile-forensics/capabilities/android-forensics (citado na pág. 99)

[65] wikipedia. (2014) Cellebrite. [*Online*]. Disponível: http://en.wikipedia.org/wiki/Cellebrite#Smartphone_forensics (citado na pág. 99)

[66] Why secure view 3 for mobile forensics? [*Online*]. Disponível: http://www.mobileforensics.com/products (citado na pág. 100)

[67] Secure view 3. [*Online*]. Disponível: http://www.secureview.us/why-secure-view-3-for-mobile-forensics (citado na pág. 100)

[68] Paraben. (2014) Device seizure. [*Online*]. Disponível: https://www.paraben.com/device-seizure.html (citado na pág. 101)

[69] Adel – android data extractor lite. [*Online*]. Disponível: http://forensics.spreitzenbarth.de/adel/ (citado na pág. 102)

[70] D. Bhatt. (2012) Howto: Use aflogical ose for logical forensics of an android device. [*Online*]. Disponível: https://santoku-linux.com/howto/howto-use-aflogical-ose-logical-forensics-android (citado na pág. 104)

[71] viaForensics Articles. (2011) viaforensics' aflogical tool is best for android forensic investigations. viaForensics. [*Online*]. Disponível: https://viaforensics.com/viaforensics-articles/viaforensics-aflogical-tool-android-forensic-investigations.html (citado na pág. 104)

Anexos

Anexo I

Título do Anexo I

Dados do Equipamento: ___

Relatorio gerado por: gorka, 20 de Junho de 2014

1. Primeiro Capitulo

1.1. Contactos no Dispositivo

ID	2	Nome:	Carlos Reis
Email			[carlos.reis@gmail.com]
Telefone			[915 828 436]
Moradas			[Rua: Fttgvg Jgfhf Hfgh Cidade: null Regiao : null Codigo: null Pais: null Tipo: 1]
Notas			[Notas:: Palhaco!g dfffcxg Fhft]
ID	4	Nome:	Cristina Francisco
Email			[cristinafrancisco@gmail.com]
Telefone			null
Moradas			[]
Notas			[Notas::]
ID	3	Nome:	Eng Joao Pedro Mateus, Engenheiro
Email			[fmat@gmail.com, fff@hotmail.com]
Telefone			[965 825 452, 852 635 844, 212 853 985]
Moradas			[Rua: Ssff Ffff Cidade: null Regiao : null Codigo: null Pais: null Tipo: 1, Rua: Ffgy Numero Xpto Cbvvng Gdvbgfgf Vghhfb Dggfh Cidade: null Regiao : null Codigo: null Pais: null Tipo: 2, Rua: Ffgy Numero Xpto Cbvvng Gdvbgfgf Vghhfb Dggfh Cidade: null Regiao : null Codigo: null Pais: null Tipo: 2, Rua: Ssff Ffff Cidade: null Regiao : null Codigo: null Pais: null Tipo: 1]
Notas			[Notas:: Isto esta quase!!!]
ID	1	Nome:	Francisco Chainho
Email			[fchainho@hotmail.com]
Telefone			[968 070 768]
Moradas			[]
Notas			[Notas::]
ID	8	Nome:	PCivil
Email			[pcivil@cm-grandola.pt]
Telefone			null
Moradas			[]
Notas			[Notas::]
ID	5	Nome:	fchainho@hotmail.com

Email			[fchainho@hotmail.com]
Telefone			null
Moradas			[]
Notas			[Notas::]
ID	7	Nome:	master@rexfile.net
Email			[master@rexfile.net]
Telefone			null
Moradas			[]
Notas			[Notas::]
ID	6	Nome:	tecnica@mbit.pt
Email			[tecnica@mbit.pt]
Telefone			null
Moradas			[]
Notas			[Notas::]

1.2. SMS no Dispositivo

Numero:	6666	O seu Lycamobile foi carregado com 5.00 Euros. O seu saldo actual e 5.00 Euros. A referencia e 2533594.
SMS	inbox	
ID	7	
Estado	1	
Time	1402136172000	
Numero:	LYCA MOBILE	O seu numero Lycamobile e 351920153153
SMS	inbox	
ID	6	
Estado	1	
Time	1402132985000	
Numero:	Web setting	Obrigado por usar o nosso servico.Sera cobrado pela utilizacao de dados apos o tempo gratuito.Ligue 1632
SMS	inbox	
ID	5	
Estado	1	
Time	1402132975000	
Numero:	Web setting	vai receber configuracoes GPRS.Por favor aceite
SMS	inbox	
ID	4	
Estado	1	
Time	1402132952000	
Numero:	+351968070768	Francisco chainho sms
SMS	inbox	
ID	3	

Estado	1	
Time	1395865287000	
Numero:	968 070 768	Francisco chainho sms
SMS	sent	
ID	2	
Estado	1	
Time	949699084735	
Numero:	212 853 985	Teste sms numero 1
SMS	sent	
ID	1	
Estado	1	
Time	949576641594	

1.3. Chamadas no Dispositivo

ID	11	date	27-Mar-2014 22:41	duration	0
type	outgoing call	Name	Francisco Chainho	number	968070768
ID	12	date	31-Mar-2014 12:43	duration	0
type	outgoing call	Name	Francisco Chainho	number	968070768
ID	13	date	31-Mar-2014 12:43	duration	0
type	outgoing call	Name	Eng Joao Pedro Mateus, Engenheiro	number	965825452
ID	14	date	01-Jan-2000 17:01	duration	44
type	outgoing call	Name		number	937022310
ID	15	date	10-Jun-2014 11:54	duration	0
type	outgoing call	Name	Carlos Reis	number	915828436
ID	16	date	10-Jun-2014 11:54	duration	0
type	outgoing call	Name	Carlos Reis	number	915828436
ID	17	date	10-Jun-2014 11:55	duration	0
type	outgoing call	Name	Francisco Chainho	number	968070768
ID	18	date	11-Jun-2014 00:23	duration	16
type	outgoing call	Name		number	16801
ID	19	date	14-Jun-2014 10:25	duration	2
type	incoming call	Name		number	920570513

1.4. Acessos ao Browse do Dispositivo

data	1402655910207	Acessos	3
Title		Google	
URL		https://www.google.com/webhp?source=android-home	
data	1402397867361	Acessos	2
Title		Google	
URL		https://www.google.pt/webhp?source=android-home&gws_rd=cr&ei=puSWU47hJ9T07AbN3YDIDw&sourceid=android-signin-promo&sa=X&ved=0CAEQwiU	
data	1402397887909	Acessos	4
Title		SAPO	
URL		http://m.sapo.pt/	
data	1402397912886	Acessos	2
Title		SAPO Desporto	
URL		http://m.desporto.sapo.pt/inicio/modal/cartoon/9c15611aaf61b66c15b44be97dafbb8c?HPho	
data	1402586441610	Acessos	1
Title		AEIOU.pt	
URL		http://www.aeiou.pt/	
data	1402586580558	Acessos	2
Title		Redirect Page	
URL		http://www.cm-grandola.pt/Paginas/redirect.aspx	
data	1402586597736	Acessos	4
Title		Yahoo	
URL		https://www.yahoo.com/	
data	1402586670362	Acessos	6
Title		www.slb - Pesquisa do Google	
URL		https://www.google.pt/search?redir_esc=&hl=pt-PT&safe=images&oe=utf-8&q=www.slb&source=android-browser-type&qsubts=1402586603066&action=devloc	
data	1402586664837	Acessos	7

Title		CheckMobile	
URL		http://m.slbenfica.pt/checkmobile.aspx	
data	1402586617225	Acessos	2
Title		SetCookie	
URL		http://m.slbenfica.pt/setcookie.aspx?mobile=1	
data	1402586660929	Acessos	2
Title		SLBenfica Mobile	
URL		http://m.slbenfica.pt/	
data	1402586658868	Acessos	3
Title		Plantel	
URL		http://m.slbenfica.pt/pt-pt/plantel.aspx	
data	1402586648800	Acessos	2
Title		Detalhe Sulejmani	
URL		http://m.slbenfica.pt/Plantel/Futebol/Detalhe/tabid/1914/PID/2409/SeID/8/Sulejmani.aspx	
data	1402586686065	Acessos	1
Title		Clix	
URL		http://www.clix.pt/	
data	1402586705403	Acessos	2
Title		Governo recupera cortes salariais que estavam em vigor no ano passado - PÚBLICO	
URL		http://www.publico.pt/economia/noticia/governo-recupera-cortes-salariais-que-estavam-em-vigor-no-ano-passado-1639641	
data		Acessos	0
Title		Google	
URL		http://www.google.com/	
data		Acessos	0
Title		Picasa	
URL		http://picasaweb.google.com/	
data		Acessos	0
Title		Yahoo!	
URL		http://www.yahoo.com/	
data		Acessos	0
Title		MSN	

URL		http://www.msn.com/	
data		Acessos	0
Title		Twitter	
URL		http://twitter.com/	
data		Acessos	0
Title		Facebook	
URL		http://www.facebook.com/	
data		Acessos	0
Title		Wikipedia	
URL		http://www.wikipedia.org/	
data		Acessos	0
Title		eBay	
URL		http://www.ebay.com/	
data		Acessos	0
Title		CNN	
URL		http://www.cnn.com/	
data		Acessos	0
Title		NY Times	
URL		http://www.nytimes.com/	
data		Acessos	0
Title		ESPN	
URL		http://espn.com/	
data		Acessos	0
Title		Amazon	
URL		http://www.amazon.com/	
data		Acessos	0
Title		Weather Channel	
URL		http://www.weather.com/	
data		Acessos	0
Title		BBC	
URL		http://www.bbc.co.uk/	

1.5. Definições do Dispositivo

Device	351554058347080
DeviceSV	01

Operador	Lycamobile
SIM Numero	8935104010017558936
Numero	unavailable
Nome Operador	Lycamobile
Codigo Pais	pt
subscriberid	268040001753893

Anexo II

Diagrama de Classes Servidor

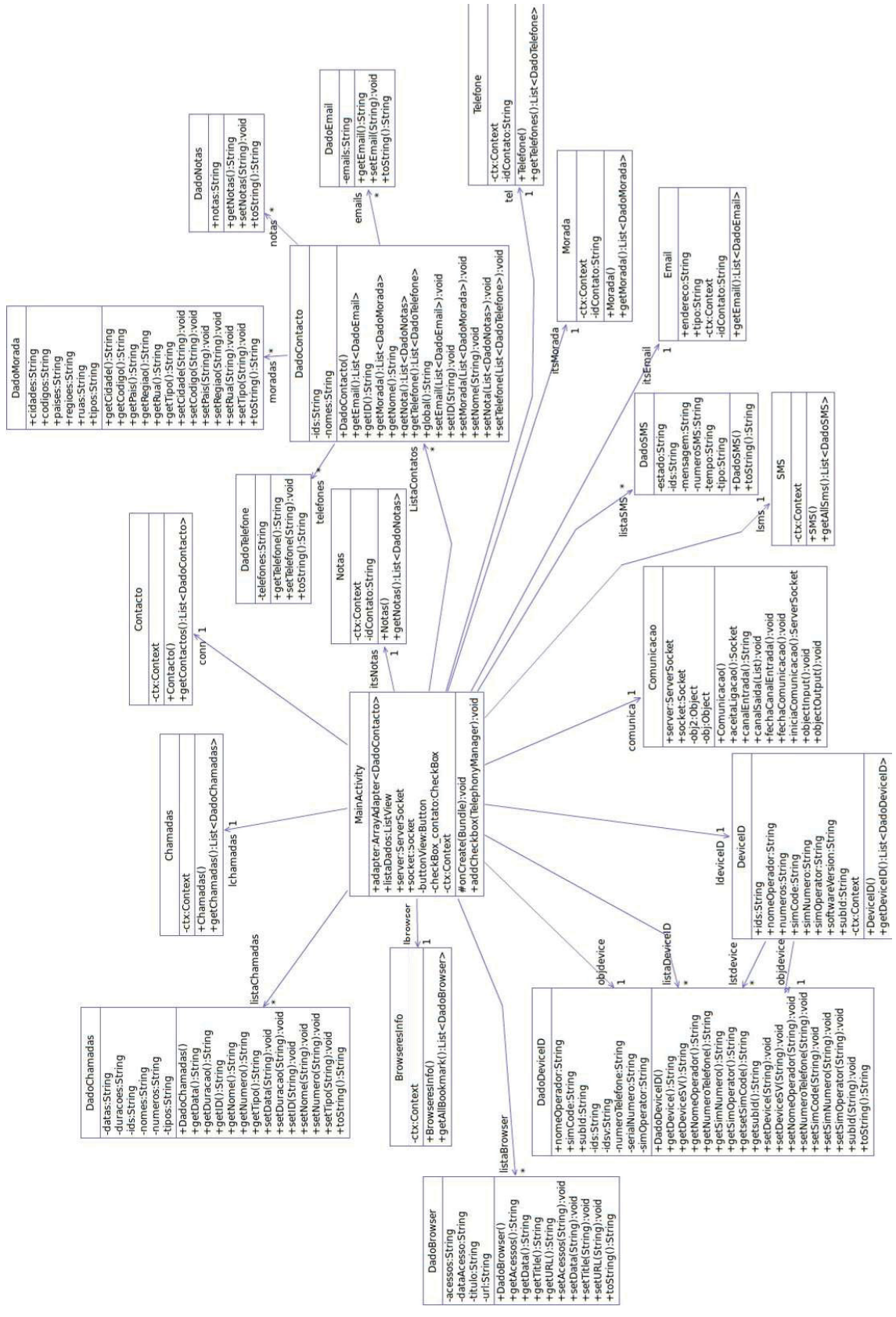


Figura II.1: Relação Classes - Servidor

Anexo III

Diagrama de Classes Cliente

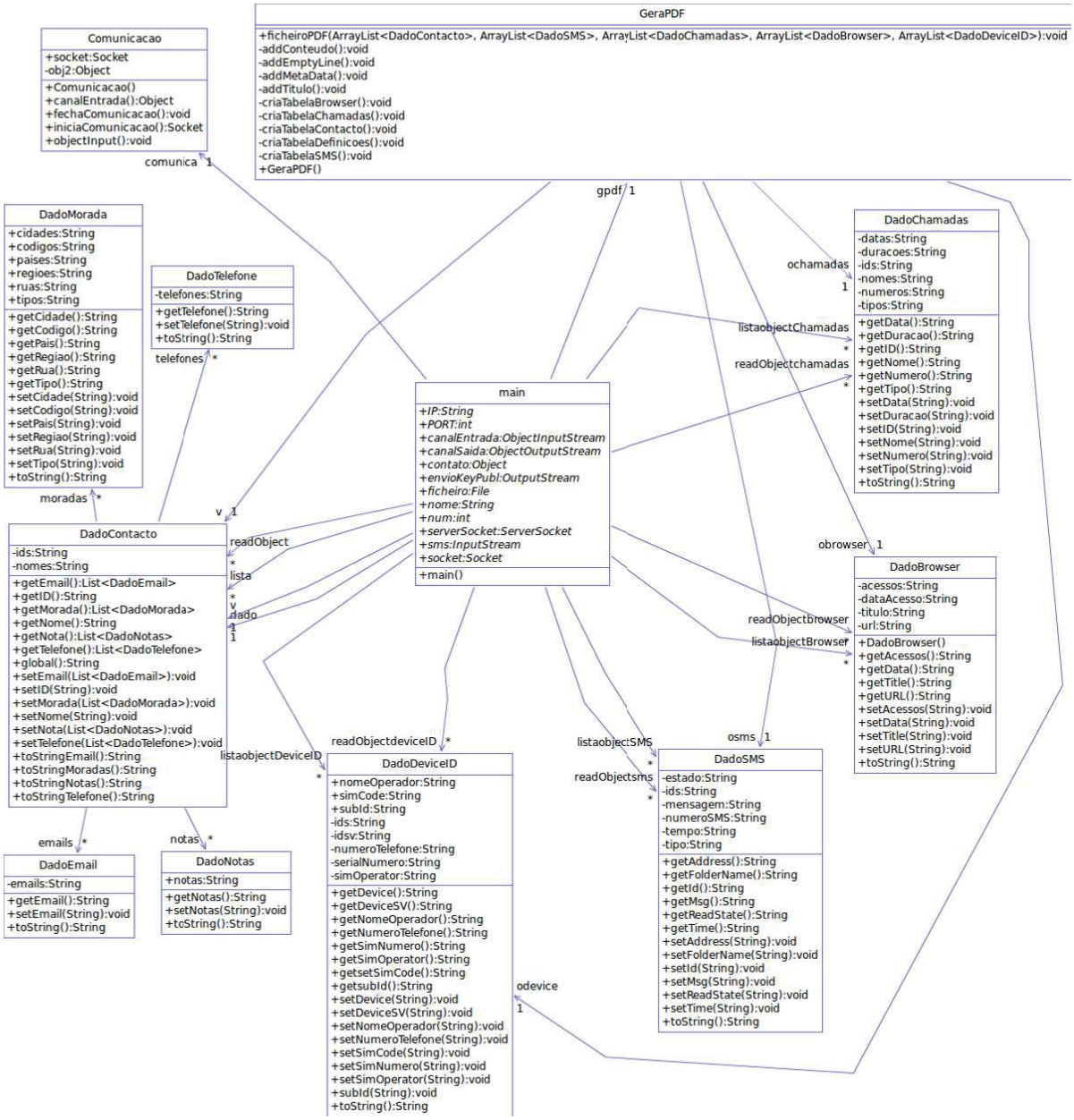


Figura III.1: Relação Classes - Cliente